LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SOLO PROGRAMADORES

ÑO VI. NÚMERO 54

UNA PUBLICACIÓN DE:
TOWER COMMUNICATIONS, S.R.L.

975 Ptas. • 5,86 € (IVA incluido)

USO DE IMÁGENES con el API JAVA 2D

MULTIMEDIA
Reconocimiento de voz.
Gestión telefónica

HERRAMIENTAS
VBScript y Active Server Pages

PROGRAMACIÓN GRÁFICA
Construcción de un polígono.
Nuevas técnicas

COMUNICACIONES
Estudio práctico del TCP

HTML DINÁMICO
Menús desplegables

CONTENIDO DEL CD-ROM

- JAVA 2 (JDK 1.2)
- NETOBJECTS FUSION 4.0
- WINGATE 3.0
- iINSTALL 3.0
- MOZILLA SOURCE CODE 5.0
- RENDERSOFT VRML EDITOR 1.7

CONTIENE NUEVA SECCIÓN DE HERRAMIENTAS Y UTILIDADES IMPRESCINDIBLES

00054
8 413042 303299

Si quieres un trabajo donde se aproveche tu talento...

PYRO Studios es una empresa española fundada en 1997 por PROEIN S.A., con el objetivo de desarrollar videojuegos innovadores para plataformas domésticas. Nuestro primer título, "**COMMANDOS: Behind Enemy Lines**", se ha convertido en el tercer juego de PC más vendido del mundo en el año 1998, con unas ventas superiores al millón de unidades.

Necesitamos personas con talento para trabajar a tiempo completo en nuestras oficinas de Madrid:

Programadores avanzados C++ (ref: PR)

Programadores en entornos orientados a objetos con diversas especializaciones (Windows, BD, COM, IA, programación gráfica 2D/3D, máquinas de estados, u otras áreas relacionadas con el desarrollo de videojuegos). Valoraremos especialmente tu autonomía, tu capacidad de organización y el dominio de las últimas técnicas en tu área de especialidad.

Grafistas 2D/3D (ref: GR)

Dibujantes/modeladores/animadores con dominio de 3D Studio MAX R2 / Adobe Photoshop. Como grafista deberás poder enseñar trabajos de la máxima calidad artística y técnica en las áreas de modelado y animación 3D, en alta o baja poligonación.

Si deseas unirte a una empresa líder en el creciente sector del entretenimiento interactivo, envíanos tu CV junto con una carta de presentación indicando todos aquellos datos y muestras de trabajo que puedan ayudarnos a valorar tu candidatura, a la siguiente dirección de correo electrónico:

seleccion@pyrostudios.com

Mantendremos referencias actualizadas de los puestos disponibles en nuestro Sitio Web:

Http://www.pyrostudios.com/trabajo.htm

...Ven a hacer los videojuegos del futuro.

PYRO STUDIOS

SÓLO PROGRAMADORES

Número 54
SÓLO PROGRAMADORES
es una publicación de
TOWER COMMUNICATIONS

Director Editor
Antonio M. Ferrer Abelló
aferrer@towercom.es
Subdirector
Oscar Rodríguez Fernández
oscarrf@towercom.es
Coordinador Técnico
Eduardo De Riquer Frutos
eriquer@towercom.es
Coordinadora de Redacción
Erika de la Riva Arnáiz
eriva@towercom.es

Colaboradores
Constantino Sánchez, Juan M. Menéndez, Jordi Agost, Javier Sanz, Adolfo Aladro, Enrique de la Lastra, Francisco M. Rosado.

Maquetación y Tratamiento de Imagen
Ana Isabel Madero Bocos

Publicidad
Olga de Oteyza (Madrid)
Tel.: (91) 661 42 11
Pepín Gallardo (Barcelona)
Tel.: (93) 213 42 29

Suscripciones
Alicia Zazo
Tel. (91) 661 42 11 Fax: (91) 661 43 86
suscrip@towercom.es

Laboratorio
Javier Amado (Jefe)
jamado@towercom.es
Servicio Técnico
Manuel Hernando
mhernando@towercom.es

Preimpresión
Indes Color
Impresión
Gráficas Muriel
Distribución
SGEL
Distribución en Argentina / Chile / Colombia / México / Venezuela
Capital: Huesca y Sanabria
Interior: D.G.P.

TOWER COMMUNICATIONS
Director General
Antonio M. Ferrer Abelló
Director Financiero
Francisco García Díaz de Liaño
Director de Producción
Carlos Peropadre
Directora Comercial
Carmina Ferrer
carmina@towercom.es
Distribución
Almiro Sanguino

Redacción, Publicidad y Administración
C/ Aragoneses, 7
28108 Pol. Ind. Alcobendas (MADRID)
Telf.: (91) 661 42 11 / Fax: (91) 661 43 86

La revista Sólo Programadores no tiene por qué estar de acuerdo con las opiniones escritas por sus colaboradores en los artículos firmados. El editor prohibe expresamente la reproducción total o parcial de los contenidos de la revista sin su autorización escrita.

Depósito legal: M-26827-1994
ISSN: 1134-4792
PRINTED IN SPAIN
COPYRIGHT 30-5-99

Editorial

Código asistido por ordenador

La verdad es que la informática y por lo tanto la programación avanza sin compasión hacia un objetivo (digo yo que alguno habrá), que no sé si es bueno o malo, pero desde luego moderno y bonito, seguro que sí. Últimamente parece que las herramientas de desarrollo ofrecen tantas facilidades, ventajas, asistentes y ayudas, que pienso que el objetivo no es otro que sustituirnos. Sí, me refiero a nosotros, a los que no hace tantos años luchábamos línea a línea, instrucción a instrucción contra la máquina, en un combate que como mínimo era de igual a igual. ¿No será que con la nueva moda de la Inteligencia Artificial el objetivo utópico de crear máquinas con cerebro haya vuelto a la actualidad?

Alguien estará pensando que igual las palabras *RAD* o *Visual* no van conmigo, pero le puedo asegurar que se equivoca, aunque a este ritmo llegará un momento en que tendremos diez asistentes diferentes para crear la famosa aplicación "Hola mundo". Todas las ayudas para mejorar la productividad y reducir tiempos y costes me parecen pocas, pero creo que no se deben confundir los términos. Dejando al margen recuerdos de codificación más o menos recientes, creo que la intención de incluir estas mejoras no debe ser que cualquiera puede programar sin problemas. Imposible. La palabra "cualquiera" no tiene ningún tipo de consideración peyorativa, o al menos ese no es mi propósito. Simplemente quiero decir que esta imposibilidad aparece en "cualquier" profesión, y que por ejemplo, yo no podría pintar como Velázquez por muchos cursos que siguiera. Me da la sensación que más o menos todo el mundo puede intentar lo que quiera, pero además de la vocación (sin duda en algunos casos mucho talento), es necesario estudiar, aprender y practicar para llevar a cabo cualquier tipo de proyecto. Siento haberme puesto tan dramático... Debe ser que con el nuevo año me estoy haciendo mayor, así que aprovecharé este momento para presentarles algo realmente joven y novedoso. Seguramente ya se habrán dado cuenta de que la revista incluye un pequeño libro práctico, cuyo objetivo no es otro que cumplir una promesa anterior. Sí, me refiero a la que hicimos cuando se produjo la separación de la revista *Sólo Linux*, y afirmamos que intentaríamos devolver el valor añadido que esta idea supuso en su momento.

Como las comparaciones son odiosas, no entraré a valorar la practicidad de las dos iniciativas y sólo me limitaré a comentar que en este caso se pretende ofrecer al lector la posibilidad de abordar un proyecto práctico de principio a fin, con la intención de desarrollar una aplicación muy concreta, sobre un tema interesante y útil (eso esperamos). Así que espero que esta parte les resulte más positiva y no piensen que hemos comenzado el año con síntomas de aburrimiento, porque nos aventuramos en un nuevo proyecto que por supuesto, sigue siendo común.

solo PROGRAMADORES

54

6 Noticias
NOVEDADES

Si quieres no perder el hilo de todo lo que se mueve en el sector de los desarrolladores no dudes en leer las noticias que te comentamos.

10
CONTENIDO DEL CD-ROM

De nuevo os invitamos a que consultéis nuestro CD-ROM. En él vais a encontrar una nueva sección de herramientas de programación imprescindibles, las fuentes que acompañan a los artículos de la revista y un montón de programas como NetObjects Fusion 4.0, Wingate 3.0, Java 2 (JDK 1.2), iInstall 3.0, Sambar Server 4.2 beta 4, JtoExe 1.0, Mozilla Source Code 5.0 y RenderSoft VRML Editor 1.7.

22 www
HTML DINÁMICO (IV). MENÚS DESPLEGABLES

Este es el cuarto artículo de la serie. A lo largo de él trataremos de exprimirle todo el jugo a HTML Dinámico a través de un novedoso y útil efecto: los menús desplegables.

28 Programación Internet
CREACIÓN DE UN BUSCADOR WEB (III)

Un robot implementado como soporte de un Buscador Web debe analizar cada página HTML que se le pase como parámetro, guardar una referencia de la misma y recuperar y volver a analizar todas las páginas web enlazadas. Veremos cómo hacerlo.

14 Programación Multimedia RECONOCIMIENTO DE VOZ (y VII)

Para concluir esta fabulosa serie de artículos sobre el reconocimiento de voz hemos decidido realizar una práctica aplicación de telefonía que nos va a permitir gestionar la entrada de datos por parte del usuario que está llamándonos por teléfono.

36
Programación Gráfica
GLIDE (III)

Continuamos nuestros balbuceos en Glide. Este mes trataremos de ir aprendiendo las normas generales para construir polígonos, representar técnicas de Antialiasing (suavizado de bordes de un polígono) y mucho más.

44
Windows 95/98/NT
WINDOWS SCRIPTING HOST (III)

A lo largo de estos artículos hemos apreciado la gran versatilidad y potencia de los scripts que podemos realizar utilizando la tecnología Windows Scripting Host. Ahora nos adentraremos en más ejemplos a fin de afianzar los conocimientos adquiridos así como introducir nuevos elementos. Nuestro objetivo es ser capaces de programar todo tipo de scripts.

50
Visual Basic
VBSCRIPT Y ACTIVE SERVER PAGES

Vamos a tratar de sacar el máximo provecho a todos nuestros conocimientos de Visual Basic para incorporarlos a Internet. Comenzamos el artículo con una visión general del lenguaje VBScript para terminar con la explicación de las páginas de servidor activo o Active Server Pages (ASP) utilizados para modificar la respuesta del servidor de páginas Web.

56 Java
TRATAMIENTO DE IMAGENES (I)

Con las primeras versiones de Java se disponía de las clases java.awt y java.awt.image para realizar un manejo de las imágenes algo reducido, pero sencillo y eficaz. Tras la aparición del API Java 2D estas posibilidades se han ampliado considerablemente, pudiéndose ahora realizar un tratamiento de imágenes mucho más potente, manteniéndose la sencillez de utilización de los comienzos.

68
Comunicaciones
ESTUDIO PRÁCTICO DEL TCP

El principal objetivo de este artículo es el de introducirnos de una manera práctica en los principios básicos del protocolo TCP. Se define protocolo como el conjunto de reglas que rigen el comportamiento de las conexiones para transferencias de datos entre dispositivos. Sin duda conseguiremos con el artículo averiguar algo más al respecto de este tema.

78
Libros
ACTUALIDAD

Os ofrecemos las últimas novedades que podéis encontrar en las librerías sobre los temas que más os interesan como son la más reciente versión de Oracle 8, un estupendo manual sobre Microsoft Visual J++ 6.0 , si te gusta más el tema de los componentes multimedia puedes obtener toda la información con la guía A fondo DirectX o si prefieres trabajar con Delphi apúntate este nombre La Biblia de Delphi3.

Comunicación con Internet desde cualquier aplicación

NETSCAPE PRESENTA EL MOTOR "GECKO", CON PRESTACIONES AVANZADAS PARA EL DESARROLLO INTERNET

Ha sido presentado el nuevo motor de navegación Gecko por la compañía Netscape Communications. Se trata de un software de nueva generación sin duda pensado para los miles de desarrolladores de Internet.

Su innovadora tecnología ha sido pensada para posibilitar la comunicación con Internet a cualquier aplicación, sistema operativo o dispositivo.

Este programa ofrece novedades y ventajas clave como son:

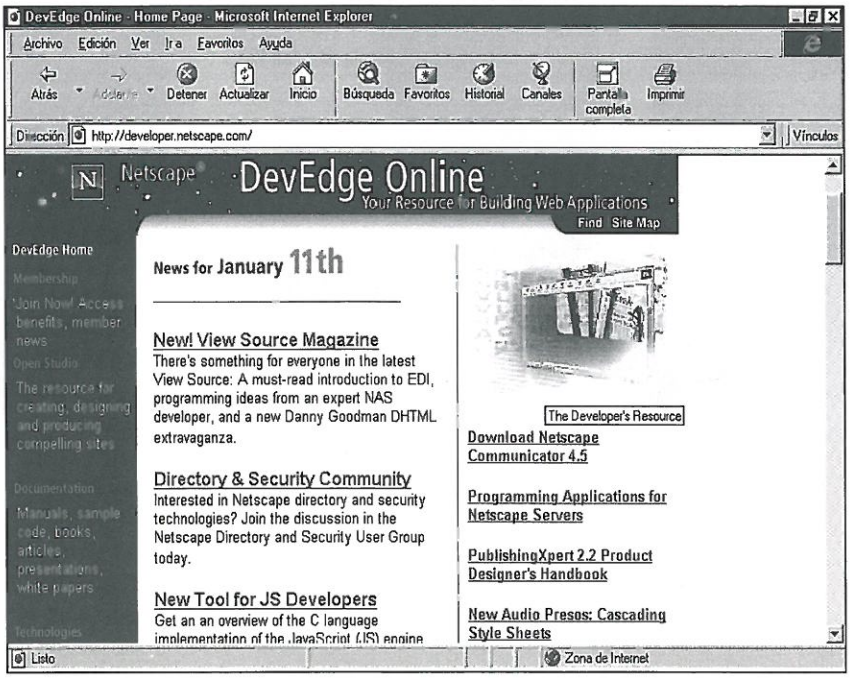
- Su tamaño, el fichero ocupa menos de 1.4 Mb.
- Su velocidad, es varias veces más rápida que las versiones anteriores.
- Estándares, dado que representa la primera y única tecnología de navegación, permite a los desarrolladores crear páginas web dinámicas y aplicaciones que funcionan en distintas plataformas.

Cabe destacar que Gecko es el primer lanzamiento de software desde la puesta en marcha de la iniciativa de código fuente abierto, impulsada por Netscape y gestionada por mozilla.org.

Además, los departamentos de tecnologías de la información corporativos disfrutarán ahora de un acceso completo al código fuente modular del motor de navegación, así como de la libertad para licenciar ese código fuente, realizar modificaciones y crear versiones personalizadas en función de sus necesidades específicas.

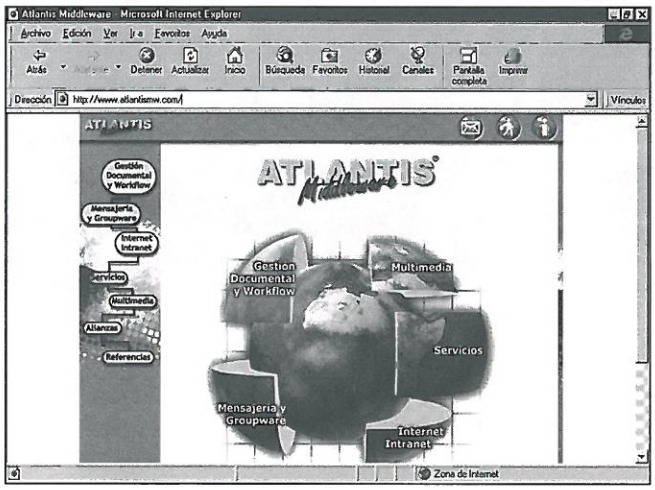
Los desarrolladores podrán personalizar el código fuente para conseguir la máxima flexibilidad, éste se puede descargar de www.mozilla.org.

También está disponible la versión de demostración de este motor que se puede descargar de la dirección http://developer.netscape.com.


DevEdge Online - Home Page - Microsoft Internet Explorer
Archivo Edición Ver Ir a Favoritos Ayuda
Atrás Detener Actualizar Inicio Búsqueda Favoritos Historial Canales Pantalla completa Imprimir
Dirección http://developer.netscape.com/
Vínculos
Netscape DevEdge Online
Your Resource for Building Web Applications
Find Site Map
DevEdge Home
News for January 11th
New! View Source Magazine
There's something for everyone in the latest View Source: A must-read introduction to EDI, programming ideas from an expert NAS developer, and a new Danny Goodman DHTML extravaganza.
Directory & Security Community
Interested in Netscape directory and security technologies? Join the discussion in the Netscape Directory and Security User Group today.
New Tool for JS Developers
The Developer's Resource
Download Netscape Communicator 4.5
Programming Applications for Netscape Servers
PublishingXpert 2.2 Product Designer's Handbook
New Audio Presos: Cascading Style Sheets
Listo
Zona de Internet

ATLANTIS MIDDELWARE LANZA MOTIVA WORKFLOW 2.0 , LA SOLUCIÓN IDÓNEA PARA EL CONTROL TOTAL DEL TRABAJO

Atlantis Middelware compañía española especializada en el diseño de tecnología orientada a objetos, presenta Motiva Workflow 2.0.



Se trata de una herramienta extremadamente flexible y potente, válida para el análisis de los procesos de trabajo y el diseño de flujos de información, que permite dar solución a los principales problemas administrativos, comerciales y ejecutivos de cualquier tipo de organización.

Motiva Workflow pone a disposición de los usuarios herramientas como son Motiva Solution Builder y Motiva Design Group, herramientas que permiten optimizar la eficiencia en la gestión de cualquier organización.

Solution Builder proporciona un conjunto de herramientas API, muy potente y fácil de usar, que simplifica la reducción de tiempo en aspectos estratégicos para la empresa, permitiendo a los desarrolladores de aplicaciones utilizarlo para la integración de sus soluciones.

Por otra parte DesignGroup facilita el acceso al control de diseño del sistema, permitiendo el depósito de documentos y el diseño de flujos de trabajo, para cubrir necesidades específicas de cada organización. Puede encontrar más información en www.atlantismw.com.

SOFTWARE AG E IBM TRABAJARÁN CONJUNTAMENTE EN EL DESARROLLO DE APLICACIONES PARA NEGOCIOS ELECTRÓNICOS BASADOS EN COMPONENTES

La compañía Software AG ha comunicado sus planes para dar soporte al desarrollo de aplicaciones basadas en componentes que puedan ser desplegadas en plataformas IBM.

Todos aquellos clientes que lo deseen podrán utilizar Bolero, la herramienta de desarrollo de aplicaciones para negocios electrónicos de reciente aparición en el mercado español, para crear componentes JavaBean que puedan ejecutarse tanto en el Component Broker como en servidores de transacciones CICS de IBM.

De este modo, lo que consiguen IBM y Software AG es mejorar de forma sustancial las opciones disponibles para los clientes que efectúan desarrollos basados en componentes para entornos empresariales.

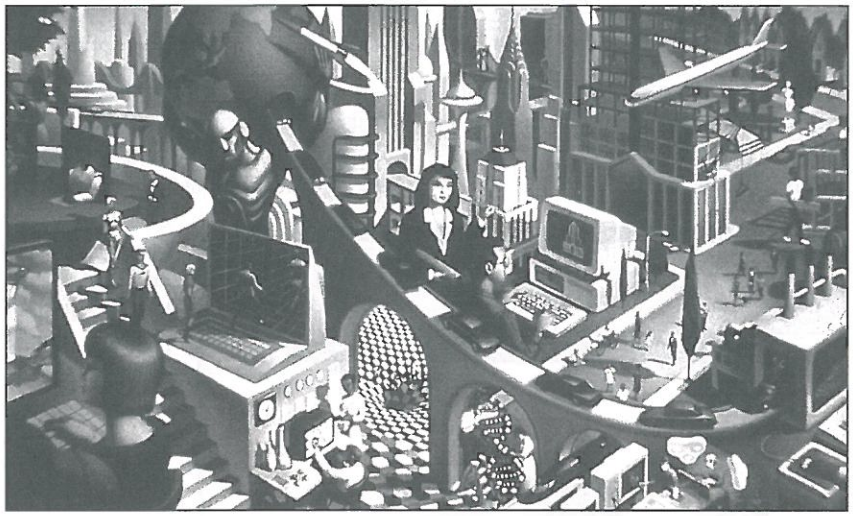
Además, de este importante acuerdo que han conseguido, estas dos compañías han anunciado también que darán soporte para la arquitectura JavaBean Enterprise para la creación y ejecución de aplicaciones de negocios electrónicos. Si lo desea puede ampliar esta información en alguna de las siguientes direcciones internet www.software.ibm.com o en www.softwareag.com.

ACUERDO ENTRE KINETIX Y DIGIMATION PARA OFRECER SERVICIOS A DESARROLLADORES

Kinetix, la división multimedia de Autodesk, ha anunciado el acuerdo al que ha llegado con la compañía Digimation. Según el acuerdo, ésta última será la publicadora preferente de Kinetix y dará a su comunidad de desarrolladores servicios de publicación.

El objetivo de esta alianza es facilitar el lanzamiento de plug-ins de terceros, a la vez que aumentar los beneficios de desarrolladores.

Los programadores que acepten publicar sus plug-ins a través de Digimation recibirán el respaldo corporativo de marketing y ventas de Kinetix que incluye campañas de promoción en sus envases, trade shows y soluciones de venta a través del canal internacional.



La información sobre dichos plug-ins también será accesible a través de webs coordinadas.

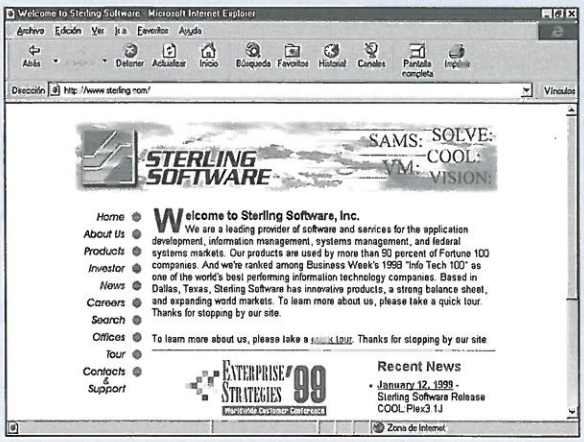
Puede encontrar información ampliada sobre este tema accediendo a www.autodesk.com.

STERLING PRESENTA LA NUEVA VERSIÓN DE SU HERRAMIENTA DE GESTIÓN DE DATOS

Sterling Software ha presentado la nueva versión de SAMS:Disk que permite a las empresas controlar los recursos y datos UNIX dentro de OS/390. Esta edición ofrece a los usuarios las ventajas de UNIX más la velocidad, seguridad, estabilidad y escalabilidad de un mainframe. SAMS:Disk, es la herramienta más amplia de gestión de datos, proporciona back up total al disco o cinta, recuperación y reporting sobre ficheros OS/390 UNIX y asegura que los datos críticos sean protegidos de la manera más adecuada.

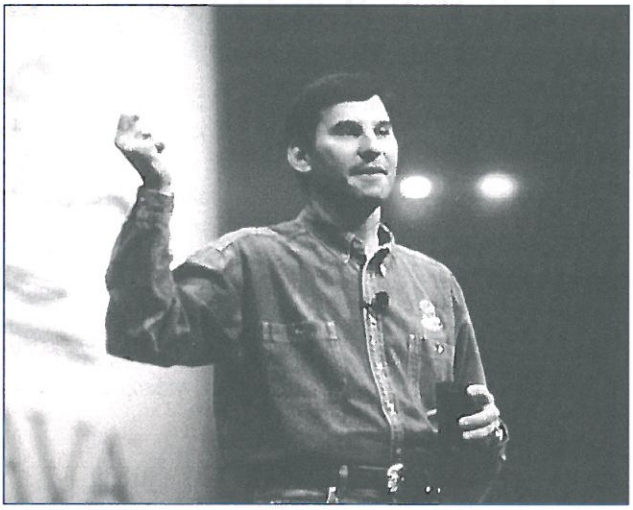
Además proporciona un interesante ahorro de tiempo y dinero, ofreciendo un control sencillo y eficaz, ya que permite al personal de almacenamiento MVS utilizar el lenguaje y los procedimientos basados en MVS para centralizar las operaciones de gestión de datos.Como resultado, se reduce el coste de formación eliminando la necesidad de conocimientos específicos de UNIX.

Si desea conocer más información acerca de esta novedosa herramienta y sobre otros productos de la compañía puede acceder a su página web a través de Internet en la dirección www.sterling.com.


Welcome to Sterling Software - Microsoft Internet Explorer
STERLING SOFTWARE
SAMS: SOLVE:
VM: COOL:
VISION:
Home
About Us
Products
Investor
News
Careers
Search
Offices
Tour
Contacts & Support
Welcome to Sterling Software, Inc.
We are a leading provider of software and services for the application development, information management, systems management, and federal systems markets. Our products are used by more than 90 percent of Fortune 100 companies. And we're ranked among Business Week's 1998 "Info Tech 100" as one of the world's best performing information technology companies. Based in Dallas, Texas, Sterling Software has innovative products, a strong balance sheet, and expanding world markets. To learn more about us, please take a quick tour. Thanks for stopping by our site.
To learn more about us, please take a quick tour. Thanks for stopping by our site
ENTERPRISE STRATEGIES '99
Recent News
January 12, 1999 - Sterling Software Release COOL:Plex3 1J
Zona de Internet

ESPAÑA, PAÍS LÍDER EN DESARROLLOS JAVA

Afirma Alan Baratz durante la presentación de Java 2 en Madrid



Sun Microsystems Ibérica ha presentado en Madrid la plataforma Java 2 hasta ahora denominada JDK 1.2. Se trata de un entorno estable, seguro y potente para el desarrollo de aplicaciones críticas en diversos sectores, destacando en nuestro país las telecomunicaciones y los servicios financieros. Así mismo la compañía hizo público el esquema general de lanzamientos para los productos de software empresarial que soportan la plataforma Java 2 que incluyen una importante línea de servidores y herramientas para el desarrollo de aplicaciones empresariales, escalables, flexibles y abiertas. Sun justificó la visita de éste por la importancia y las posibilidades de futuro del mercado español.

En un largo turno de preguntas y respuestas estos son algunos de los aspectos que aclaró para nosotros el Presidente de Java Software durante el acto de presentación de la nueva plataforma.

SP. ¿Esta versión de Java 2 es una versión definitiva?
AB. Efectivamente. Se trata de una versión completa y estable que incluye todos los APIS necesarios para el desarrollo de aplicaciones, de modo que podemos garantizar que por lo menos en un año y medio no habrá una nueva versión.

SP. Algunos profesionales no han hecho llegar su inquietud ante las incompatibilidades detectadas entre las plataformas Java 1 y Java 2. Podría aclararnos en que medida es esto cierto.
AB. La evolución de la informática implica la imposibilidad de mantener la compatibilidad si se quiere conseguir un determinado nivel de prestaciones por lo que ciertamente existen algunas incompatibilidades en aplicaciones desarrolladas bajo versiones anteriores al JDK 1.1.6 con las desarrolladas con el nuevo Java 2. Para aplicaciones desarrolladas con el JDK 1.1.6 sólo existen ciertos puntos de incompatibilidades con aspectos menores fácilmente subsanables. Aprovechamos las páginas de Sólo Programadores para decirles a todos los desarrolladores que detecten alguno de estos problemas que estamos a su disposición para ayudarles en la resolución de sus dudas y problemas técnicos.

SP. ¿Por qué conserva Sun el control del estándar Pure 100% Java y cuál es el proceso de homologación de una aplicación?
AB. En este momento ofrecemos al mercado la plataforma de desarrollo de aplicaciones empresariales más abierta posible de modo que el kit de desarrollo está disponible en nuestro Web y se está distribuyendo también a través de algunos medios de comunicación especializados. Sun conserva el control del estándar para evitar lo que sucedió con Unix, ya que al ser totalmente libre acabó con múltiples versiones sin conseguir toda la expansión de mercado que podía haber alcanzado.

Por otra parte se tratan de evitar duplicidades inútiles en el desarrollo del estándar Java y conseguir que los desarrolladores unan sus fuerzas, no luchen entre si y se cree un verdadero estándar en el que todo el mundo se beneficie de las mejoras.

La homologación de las aplicaciones la realiza Sun de forma gratuita y las empresas tan sólo pagan un royaltie (muy bajo) cuando las venden. El importe de este royaltie puede variar entre medio dólar y 5 dólares dependiendo del tipo de aplicación que sea.

SP. ¿No se convertirá entonces Sun, gracias al control del estándar Java, en el nuevo Microsoft?
AB. No, en absoluto. Con el nuevo modelo de licencias que acabo de explicar Sun tan sólo persigue garantizar la universalidad de la plataforma Java. Por esta razón el kit de desarrollo está a disposición de cualquiera, sin que deba pagar nada por ello.

HERRAMIENTAS DE PROGRAMACIÓN

ENTORNOS DE DESARROLLO

JAVA DEVELOPMENT KIT 1.2 DE SUN

La herramienta de desarrollo estándar de la plataforma Java 2. Contiene todo el software y las herramientas necesarias para compilar, testear y ejecutar applets y aplicaciones desarrolladas en lenguaje Java. La versión 1.2 proporciona nueva funcionalidades, como Java 2D, accesibilidad, drag and drop, colecciones, métodos de introducción de código, identificación, IDL, JVMDI, y muchas más.

68000 EDITOR, ASSEMBLER AND SIMULATOR 3.0

Entorno para escribir y compilar ensamblador para procesadores Motorola 68000. Permite escribir y compilar programas en lenguaje ensamblador para el procesador Motorola 68000. El programa está diseñado para que se pueda editar, compilar y simular programas 68000 bajo entorno Windows.

AUTOPLAY DREAM KIT

Entorno para el desarrollo de aplicaciones AutoPlay. Permite escribir aplicaciones 16 y 32 bits que lleven incorporada la función AutoPlay. Permite lanzar imágenes bitmap y sonidos, capturar mensajes tras los movimientos de ratón y "pintar" sobre la pantalla imágenes y texto. .

MACROMEDIA AUTHORWARE 5.0 ATTAIN

El entorno estándar para la creación de aplicaciones multimedia profesionales. Es un entorno de desarrollo visual que proporciona gran capacidad para la creación de aplicaciones multimedia profesionales.

NETSCAPE GECKO (DEVELOPER PREVIEW)

Nueva tecnología de navegación para comunicar una aplicación con Internet. Su revolucionaria tecnología ha sido diseñada para posibilitar la comunicación con Internet a cualquier aplicación, sistema operativo o dispositivo. Dado que Gecko representa la primera y única tecnología de navegación que implementa íntegramente los últimos estándares de W3C.

MICROSOFT WINHELP 3 COMPILER 5.05

La herramienta indispensable de Microsoft para compilar ficheros de ayuda. Compilador necesario para la creación de ficheros .HLP de ayuda para adjuntar a desarrollos para Windows.

STANDARD PERL 5.004

Un entorno Win32 para el desarrollo y programación en Perl. Cuenta con practicamente todo lo que se necesita para utilizar Perl en las plataformas de 32 bits. También cuenta con todo lo que se necesita para desarrollar, analizar e instalar componentes PERL de manera sencilla. Para ello utiliza la extensión estándar MakeMaker.

POLYMEDIA SDK 2.0

Paquete integrado de controles ActiveX para desarrollar multimedia usando DirectX. Polimedia SDK elimina la necesidad de encontrarse con las complicaciones de trabajar con las estructuras internas de desarrollo de DirectX. Para ello ofrece el acceso rápido y ágil a los componentes que ofrecen la funcionalidad y potencia de DirectX. Posibilita desarrollar operaciones gráficas potentes y veloces a través de objetos multimedia incluyendo vídeo, animación, texto y gráficos.

RENDERSOFT VRML EDITOR 1.7

GUI que facilita el desarrollo de mundos virtuales para páginas Web. Permite crear, editar y manipular un mundo interactivo 3D VRML en tiempo real. Aparte de las transformaciones de objetos como movimiento, rotaciones y escalas, también incluye un soporte de efectos de luz, texturas y materiales.

LENGUAJES

VISUAL BASIC

AXGRID 2.0.16

Es una versión mejorada del FlexGrid incluido con Visual Basic. Añade diferentes mejoras en las capacidades de edición incluyendo la habilidad de añadir automáticamente una nueva entrada cuando el usuario intente editar la última fila en blanco.

ACTIVEZIPPER PRO 1.0F

Pequeño y rápido control de VB que permite la compresión y descompresión de archivos. Trabaja con el formato ZIP y

soporta característica de trabajo con un archivos múltiples, protección de contraseñas, chequeo CRC, encriptación de ficheros, etc.

ERROR MAGICIAN 4.0.0

Es un detector automático de errores globales en código Visual Basic. En modo automático, muestra al usuario en tiempo real los errores cometidos. Además se puede compilar utilizando la librería DLL ActiveX para detectar errores y que estos se puedan solventar.

JAVA RIADABUTTON 1.0

Creación automática de botones de navegación Web basados en Java. RiadaButton permite crear botones de navegación Web basados en Java, incorporar audio, animación, herramientas de ayuda, texto dinámico, etc. Además va más allá que los botones estándar Web, permitiendo diseñar y generar botones de carga rápida que obedecen al ratón.

JTOEXE 1.0

Utilidad para crear ficheros ejecutables de programas Java. Una aplicación extremadamente útil tanto para usuarios como para desarrolladores. JtoExe permite elegir una aplicación JAVA e guardarla como un fichero .EXE de Windows, permitiendo ejecutar aplicaciones JAVA con un simple clic de ratón en cualquier equipo.

JAVA EXPLORATION TOOLS 1.13

Generador de árboles visuales a partir de clases Java. JAVA Exploration Tools es un generador de árboles visuales a partir de clases JAVA.

MICROSOFT JAVA VIRTUAL MACHINE

La última actualización de la Microsoft Virtual Machine for Java. Esta nueva versión resuelve algunos bugs de las anteriores distribuciones e incluye como novedad el denominado Java Native Interface (JNI), una tecnología para la creación de aplicaciones Java específicas para las distintas plataformas.

HTMLNETOBJECTS FUSION 4.0

Herramienta profesional para la gestión visual de sites Web. Se trata de una de las herramientas con más prestigio entre los Webmasters. Permite crear y gestionar página Web dinámicas a través de un editor visual que muestra la estructura del sitio para tener un control total de su funcionamiento y apariencia.

ULEAD WEBRAZOR PRO 1.0

Colección de herramientas profesionales para el diseño gráfico Web. El paquete está compuesto por GIF Animator 3, SmartSaver Pro, Cool 3D 2 y Photo Explorer 4.2. El primero permite crear impactantes GIF animados. SmartSaver posibilita la manipulación y optimización de imágenes para su uso.

DOCFATHER PROFESSIONAL 2.2

Creación de índices de búsqueda y navegación para proyectos Web. Permite crear los índices en sites, documentos HTML en general e incluso en ficheros de texto. Cuenta con un visor HTML que muestra los índices y que incluye botones de navegación, soporte para impresión, gestión de favoritos, etc.

WEBCOMPILER 98 1.10

Creación de autoejecutables con sites Web. Es capaz de crear un fichero .EXE de las páginas HTML y crear con ellas un archivo navegable y modificable en cualquier equipo, lo que pemite que el código sea fácilmente transportables. El interfaz ofrece links y búsquedas. Ofrece restricción de acceso a las páginas para asegurar la privacidad.

NAVROAD OFFLINE HTML BROWSER 2.20

Navegador diseñado especialmente para la visualización off-line de páginas HTML. Es un pequeño y eficiente navegador diseñado para la visualización e impresión off-line (en local) de archivos HTML. Puede ser utilizado por cualquiera que necesite visualizar archivos HTML mientras no está conectado a Internet.

GASPY GOLD 1.6

Editor especial sólo para desarrolladores Web. Es un Freeware que soporta ficheros HTML, ASP, JAVA, JavaScript, y de texto puro. Proporciona sintaxis configurable de colores y tipos de ficheros; permite salvar y exportar plantillas de códigos; incluye autocorrectores para los comandos más utilizados.

OTROS

AHM TRITON TOOLS 98

Incluye componentes para realzar o mejorar menús, E-mail, controles background, componentes invisibles de sistema y mucho más. Existen limitaciones en esta versión, los componentes de la trial caducan o vencen, situación que se comunica mediante la solicitud de los datos de registro. Sólo los componentes freeware continuarán trabajando sin necesidad de que se visualice un cuadro de diálogo que solicita el registro por parte del usuario.

CODE NAVIGATOR 2.01 FOR C++

Potente sistema para la creación, edición y navegación con código C++. Permite acelerar el desarrollo de software ofreciendo una edición visual del código y sus atributos, creación de código con arrastrar y soltar y un software automático incorporado diseña dibujos o esquemas escritos en C++ que pueden ser aplicados medianet drag&drop.

LISP EXPLORATION TOOLS 1.55

Es una función LISP para el tratamiento y creación de estructuras de árbol. Analiza los códigos fuente LISP para proporcionar una representación inter-

na de la función con todas las relaciones y jerarquías. Incluye opciones para personalización de la salida, que puede ser en formato HTML o RTF.

RS MAGIC SLIDER 1.0.3

Control ActiveX que permite crear barras de navegación de imagen futurista. Permite crear diseños muy atractivos en los que pueden intervenir imágenes de fondo, tipografías específicas y cualquier bitmap.

HERRAMIENTAS/ UTILIDADES

DISKEEPER LITE 1.1.137 PARA WINDOWS NT

Un programa defragmentador de discos específico para Windows NT. Puede defragmentar volúmenes FAT y NTFS. Es muy seguro y realiza varios pases de optimización durante cada sesión, lo que le permite defragmentar archivos complicados y crear más espacio libre. .

GETRIGHT 3.31

Utilidad para la automatización de tareas y bajada de ficheros automática. GetRight es una utilidad para la bajada de ficheros automática que permite establecer los horarios a los que el usuario desea realizar el download y que permite recuperar los archivos cuando se producen errores durante la conexión. Permite interrumpir el proceso de descarga y reiniciarlo de nuevo por completo desde el punto y sólo recuperar la parte restante del fichero.

IINSTALL 3.0

Sistema freeware para el desarrollo de procesos de instalación. ilnstall es un software de compresión y un sistema de instalación y desinstalación de archivos para la distribución profesional de programas.

RESOURCE-GRABBER 1.80

Recupera información, para su reutilización, de cualquier tipo de archivo. Explora y extrae ficheros bitmap, imágenes de botones, iconos y cursores de programas y librerías DLL. En vez de crear una y otra vez iconos o botones para las aplicaciones, simplemente basta con reutilizar imágenes que ya están en otros desarrollos. Resource Grabber los extrae de sus ubicaciones ocultas en archivos DLL y EXE y lo recupera como bitmaps estándar. Puede incluso extraer fichero AVI, imágenes JPG incorporadas.

ARITECH INI FILE EDITOR 1.71

Editor freeware especializado en la modificación de ficheros .INI. Es un editor de texto con un potente paquete de características para la modificación de características de los archivos .INI específicos de Windows.

WINEDIT 321-99B

Editor muy potente diseñado sólo para programadores. Todas las opciones que necesita un profesional. Incluye características de buscar y reemplazar, fuentes ajustables, cabeceras, numeración de líneas, impresión a doble cara, apertura de múltiples archivos, deshacer y rehacer, sintáxis de código con color configurable, plantillas, etc.

INSTALL-US PRO 3.50

Kit de herramientas para la creación de sistemas de instalación profesionales. Se pueden crear sistemas simples en sólo unos minutos utilizando un asistente, el programa proporciona un control a fondo sobre practicamente cada tipo de parámetro de las distintas instalaciones. Proporciona un completo control sobre el registro de Windows, ejecución e instalación.

REDES

LOCALES

SEE4VB 2.1

Servidor de E-mail SMTP/POP3 para Visual Basic. La utilidad consiste en una librería .DLL de 16 bits y otra de 32 bits que permiten a cualquier aplicación desarrollada en Visual Basic o Visual Basic para aplicaciones enviar y recibir E-mail a través de Internet. Incluye la posibilidad de trabajar con ficheros adjuntos.

WINGATE 3.0

Para conectar un entorno LAN a Internet utilizando una sóla conexión. Cuando se instala WinGate, se debe registrar online y seleccionar o no si se está instalando versiones Home, Standard o Pro. La versión Home permite compartir una conexión Internet con tres o cuatro ordenadores, y ofrece la mayoría de características de las más avanzadas versiones.

DISTRIBUIDAS

THE INTERCOM SYSTEM 1.0

Solución para crear aplicaciones con opciones avanzadas para la comunicación en redes. El sistema es útil para desarrollar algunas aplicaciones multiusuario que permitan la compartición de recursos, como groupware o incluso juegos en red.

SAMBAR SERVER 4.2 BETA 4

Es un servidor HTTP freeware de fácil uso y sencilla configuración. Es de uso libre e incluye capacidades multi-threaded, un motor de búsqueda, generación de páginas DHTML, SMTP, servicios proxy POP3 e IMAP4, servidor FTP interno, soporte CGI/1.1 (incluyendo Perl 5), análisis a través de logs en HTTP y FTP, etc.

OTRAS

MABRY FTP/X 2.00.000

Control ActiveX y objeto COM que proporciona acceso a un servidor FTP. Obtiene el listado completo de los ficheros existentes en el servidor y los puede guardar en un ADO Recordset. Trabaja con dos modos diferentes de operación: síncrono o asíncrono.

PING PLOTTER 2.03

Es una utilidad traceadora gráfica muy rápida. Basta con teclear la dirección sobre la que se necesita la información y a través de Ping Plotter aparecerá una vista gráfica en la que se muestra la información técnica sobre el tiempo de respuesta del origen y el destinatario.

DOCUMENTACIÓN Y TUTORIALES

REGISTRYFAQ, PREVIEW RELEASE 1.0

Es una excelente fuente de información relacionada con el Registro de Windows. Incluye los consejos y trucos avanzados. Se presenta en un documento PDF de Adobe Acrobat que incluye más de 130 páginas con más de 250 temas individuales. Además de tratar la composición del Registro, cubre algunos de los temas más oscuros y que aún no han sido tratados en manuales de Microsoft.

JAVA DEVELOPMENT KIT DOCUMENTATION 1.2

Documentación técnica de Sun sobre el nuevo JDK 1.2. Una joya que incluye las últimas notas técnicas, las especificaciones de todos los APIs, documentación de las herramientas, una guía de las nuevas características del kit de desarrollo y varios ejemplos de programación en Java.

DIPPY BIRD'S JAVASCRIPT DOCUMENTATION

Ofrece la Guía de JavaScript de Netscape y otras guías relacionadas con el tema en formato ayuda de Windows. Incluye también obras como JavaScript Authoring Guide, LiveWire Developer's Guide, Dynamic Style Sheets y Positioning HTML Elements. A todas estas guías se puede acceder fácilmente desde el HLP de Windows ofreciendo una rápida navegación.

MOZILLA SOURCE CODE 5.0 (BUILD 12/11/98)

Código fuente completo en C/C++ de Netscape Communicator. Netscape ha hecho todas las versiones de Navigator y Communicator gratuitas y además ha liberado sus códigos fuente para posibilitar la implementación de nuevas capacidades, aportadas por los programadores, en futuras versiones de su software.

FUENTES DE LOS ARTÍCULOS

HTML Dinámico (IV)
Ejemplos de algunos efectos realizados con HTML Dinámico ./program/DHTML4/

Reconocimiento de voz (y VII)
Los fuentes de la práctica propuesta en el artículo ./program/RVOZ7/

Estudio práctico del TCP
Prácticas propuestas en el artículo ./program/COMUNICA/

Creación de un buscador Web (III)
Los fuentes de la práctica propuesta en el artículo ./program/BUSCA3/

Windows Scripting Host (III)
Los fuentes de la práctica propuesta en el artículo ./program/WSH3/

IMPRESCINDIBLES

ANTIVIRUS

McAfee VirusScan
Panda Antivirus Platinum

GRAFICOS

The Best Icons
LView Pro Image Processor 2.1
Paint Shop Pro 5.01
ThumbsPlus 3.30s
Xara 3D 3

INTERNET

AutoWinNet 5.5 Beta 2
CuentaPasos 3.51
CuteFTP 2.6
Eudora Light 3.0.6
Go!Zilla 3.3
HomeSite 4.0
mIRC 5.5
URL Organizer 2.0
WebZIP 2.50

MULTIMEDIA

GoldWave 4.02
WinAmp 2.05

NAVEGADORES

Netscape 4.08
Opera 3.51

RUNTIMES

Runtimes de Visual Basic

UTILS

Adobe Acrobat Reader 3.01
Babylon Translator
CDR Win
Day Time Organizer 2000
DirectX 6.0
Fix 2000
Cybermedia Uninstaller
Windows Commander 3.53
Win32s
WinZip 7.0

Reconocimiento de voz (y VII)

Constantino Sánchez Ballesteros (constantino@nexo.es)

Concluimos esta interesantísima serie de artículos sobre el tema del reconocimiento de voz. Para ello, realizaremos una importante aplicación de telefonía que nos va a permitir la gestión de la entrada de datos por parte del usuario que está llamándonos por teléfono.

NOTAS SOBRE TELEFONÍA

Las aplicaciones de telefonía pueden ejecutarse de dos modos:

1. Modo emulación.
2. Modo telefónico.

El primer modo utiliza el micrófono y los altavoces en lugar del teléfono, mientras que el segundo utiliza la *API TAPI* para modems con soporte de voz. La diferencia principal entre el modo emulación y el modo telefónico es ninguna.

Para utilizar un *Voice modem* debemos tener una versión reciente de nuestro *driver* del módem. Muchos *Voice modems* utilizan el *driver* llamado **UnimodemV** que puede encontrarse en la página web de *Microsoft*.

Los *Voice modems* frecuentemente no poseen la calidad de sonido o el tiempo de respuesta obtenidos con el emulador. Si la calidad es demasiado pobre, muy a nuestro pesar, tendremos que comprar un módem de mayor calidad o una tarjeta multilínea en el caso de que nuestras necesidades estén orientadas hacia un terreno profesional.

Las tarjetas multilínea son ejecutadas con múltiples tareas al mismo tiempo. El depurador de *Visual Basic* no puede ser utilizado con *applets* multitarea, por lo que tendremos que depurar línea por línea o trabajar con el emulador. Para ejecutar múltiples líneas deberemos compilar la aplicación y ejecutarla fuera del entorno de *Visual Basic*.

NOTAS DEL PROGRAMA

Este *applet* es un modelo de cómo efectuar registros de programas *shareware* a través del teléfono. La idea consiste en que la versión *shareware* de un programa termina en una determinada fecha o tiene características no funcionales, y un usuario debe registrar el programa para obtener la versión completa del mismo. Para efectuar esto de forma robusta, el programa *shareware* genera un *password* que identifica de forma única el ordenador sobre el que se está ejecutando (como por ejemplo, el número de serie de la unidad de disco duro).

El modo emulación utiliza el micrófono y los altavoces en lugar del teléfono

De este modo, la llave de registro puede ser encriptada con esta importante información, asimismo también será guardada en el registro de *Windows* consiguiendo de esta manera que la versión registrada del programa sea incopiable. (¿!).

Advertencia al Programador

La versión beta del reconocimiento continuo de la API Speech no funciona en modo multitarea. Debemos establecer el engine command and control si queremos probarlo sobre una tarjeta multilínea.

El proyecto que veremos en este artículo utiliza Text-To-Speech (texto hablado) para reproducir todas las palabras. Para una aplicación real podríamos utilizar palabras pregrabadas de una voz humana en ficheros WAV. Afortunadamente, la API tiene el programa editor de listas de WAV y la interfaz WaveAddFromList para hacer las cosas más fáciles al programador.

Este ejemplo es simplemente eso: un ejemplo. En él se demuestran muchas características que podemos utilizar para almacenar información como el números de la tarjeta de crédito de la persona que llama. Por supuesto, queda a expensas del programador realizar el código necesario para validar y cargar dicha tarjeta.

Las tarjetas multilínea son ejecutadas con múltiples tareas al mismo tiempo

BOTÓN START MONITORING DEL FORMULARIO

Este botón se utiliza para dar comienzo a la sesión del programa monitorizando los datos que nos vayan llegando. El código asociado al mismo es el siguiente:

```
Private Sub Command1_Click()
 Form1.Visible = False 'esconde el formulario
     principal
 If (Option1.Value) Then
  phone1.Initialized = 1  'ejecuta el emulador
 Else
  phone1.Initialized = 2  'establecemos cone-
     xión con TAPI
 End If

 phone1.CallDialog        'Ejecutamos una
     tarea por cada línea de teléfono.
 Form1.Visible = True     'activa el formulario
     principal
End Sub
```

BOTÓN SELECTION ENGINES... DEL FORMULARIO

Este botón se utiliza para seleccionar los *engines* de reconocimiento que queramos utilizar en nuestra aplicación. El código asociado a este botón es el que aparece a continuación y *ChooseEngineDialog* permite al usuario establecer el *engine* de texto hablado seleccionado, así como el *engine* de comandos de voz.

```
Private Sub Command3_Click()
 phone1.ChooseEngineDialog
End Sub

'Botón 'Exit' del formulario
Private Sub Command2_Click()
End
End Sub
```

En el siguiente procedimiento se establece la clave del programa para cada línea de teléfono utilizada. La variable especial **lineID** identifica qué línea/tarea se está utilizando en un momento determinado. Todas las variables utilizadas en esta rutina deben ser locales a la misma. No podemos modificar variables globales desde esta rutina porque se puede estar ejecutando sobre múltiples tareas asíncronas de forma simultánea:

```
Private Sub phone1_DoPhoneCall(ByVal lineID
     As Long)

 Dim sPassKey As String
 Dim CFirstFour As String
 Dim CNextSix As String
 Dim CLastFive As String
 Dim CExpDate As String
 Dim result As Long
 Dim s As String
 Dim ss As String
 Dim sDTMF As String
```

La siguiente línea debería ser activada después de utilizar el editor de listas de *WAV* para cambiar los textos hablados a voces humanas establecidas en archivos de este tipo. Debemos asegurarnos de establecer correctamente el directorio de trabajo:

```
phone1.WaveAddFromListFile lineID, "c:\direc-
     tory\wavemap.txt"
```

A continuación damos la bienvenida al usuario que llama y le invitamos a registrarse. Para ello utilizamos el método *Speak* del que podemos encontrar información detallada en la tabla 1.

```
phone1.Speak lineID, "Bienvenido al centro de
     registro automático de programas Share-
     ware. Es necesaria la clave del programa
     para registrar la versión Shareware.
     Puede obtener la clave de la versión
     shareware del programa eligiendo Pass-
     word desde el menú de registro. Por
     favor, hable claramente y espere a que
     termine de hablar antes de que usted
     conteste las preguntas."

PassKey:
```

Llama a la función de ayuda para obtener números (ver más adelante).

```
 s = GetNumber(lineID, 4, "Por favor, diga los
     4 números de la clave.", _
                "Qué es la clave?", _
                "", _
                "")
 If (s = "") Then
  GoTo done
```

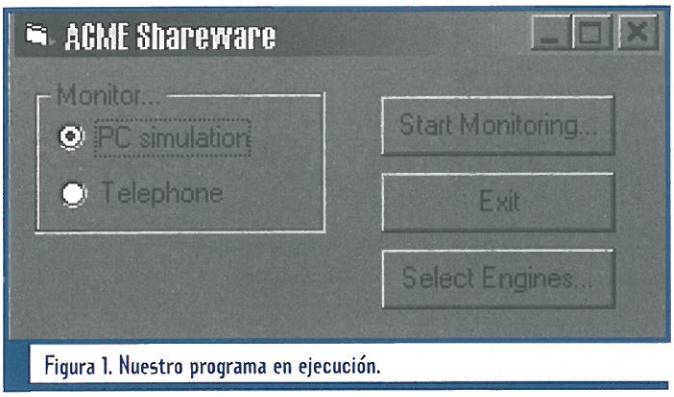


Figura 1. Nuestro programa en ejecución.

```
End If

phone1.Speak lineID, "tu clave es "
phone1.Speak lineID, s

sPassKey = s
```

Siempre debemos verificar la información, ya que los errores son comunes. Nosotros utilizaremos la versión *string* del control *YesNo* debido a que las preguntas son cortas.

NOTA: *debemos agregar nuevas líneas por cada frase.*

En la tabla 2 se puede ver información relativa al método *YesNoFromString*:

```
phone1.YesNoFromString lineID, "[Prompts]"
    + vbNewLine + _
                "Main=Es esto correc-
    to?" + vbNewLine + _
                "Where=obteniendo
    Password" + vbNewLine + _
                "[Settings]" + vbNewLi-
    ne + _
                "CanGoBack=1" +
    vbNewLine, result, s
  If (result < 0) Then
    GoTo done
  End If
  If (result <> 1) Then
   GoTo PassKey
  End If
```

Obtenemos el número de la tarjeta de crédito (función de ayuda *GetNumber*). Por supuesto, en el mundo real necesitaríamos obtener la marca de fábrica de la tarjeta y utilizar el formato correcto (por ejemplo, *VISA*, *6000*, etc.).

```
  phone1.Speak lineID, "Ahora necesitamos su número de tarjeta de crédito."

firstfourcredit:

s = GetNumber(lineID, 4, "Por favor, diga los primeros 4 dígitos de su tarjeta de crédito ", _"Qué son los primeros 4 dígitos?", _"está introduciendo los primeros 4 dígitos de su tarjeta", _"Dígame su número de tarjeta de crédito")
  If (s = "") Then
   GoTo done
  End If

  phone1.Speak lineID, "Los primeros 4 dígitos de su tarjeta de crédito son "
  phone1.Speak lineID, s
  CFirstFour = s
```

Como siempre, verificamos los datos obtenidos:

```
  phone1.YesNoFromString lineID, "[Prompts]"
      + vbNewLine + _
    "Main=Es correcto?" + vbNewLine + _
    "where=Obteniendo los 4 primeros números" + vbNewLine + _
    "[Settings]" + vbNewLine + _
    "CanGoBack=1" + vbNewLine, result, s
  If (result < 0) Then
    GoTo done
   End If

  If (result <> 1) Then
   GoTo firstfourcredit
  End If

nextsixcredit:
  s = GetNumber(lineID, 6, "Por favor, introduzca los siguientes 6 dígitos del número de su tarjeta de crédito ", _ "Qué son los siguientes 6 dígitos?", _
    "está introduciendo los siguientes 6 dígitos del número de su tarjeta de crédito. ", _
    "Dígame su número de tarjeta de crédito")
  If (s = "") Then
   GoTo done
  End If
```

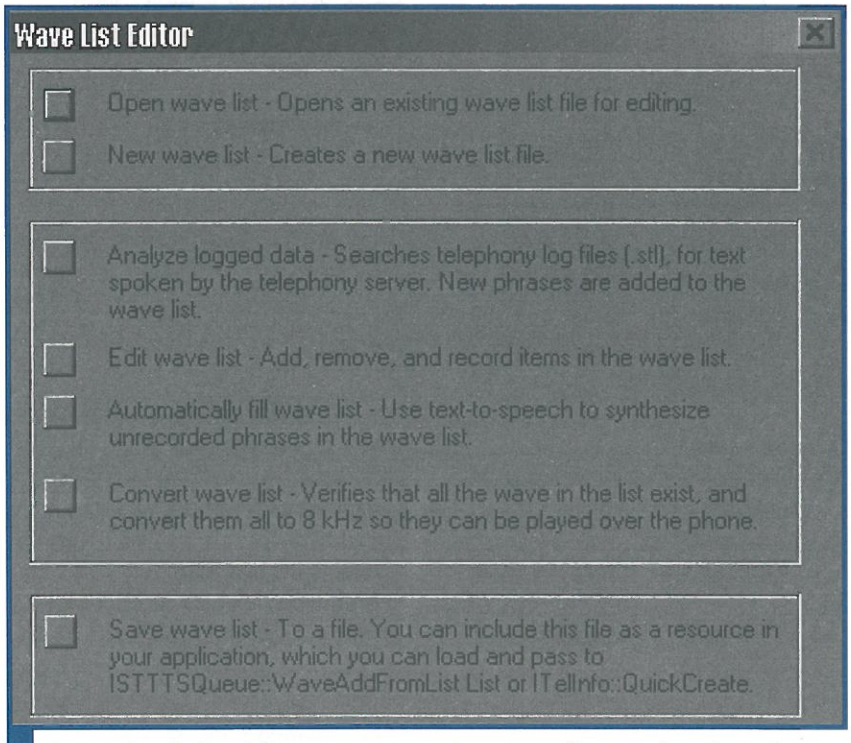

Wave List Editor
Open wave list - Opens an existing wave list file for editing.
New wave list - Creates a new wave list file.
Analyze logged data - Searches telephony log files (.stl), for text spoken by the telephony server. New phrases are added to the wave list.
Edit wave list - Add, remove, and record items in the wave list.
Automatically fill wave list - Use text-to-speech to synthesize unrecorded phrases in the wave list.
Convert wave list - Verifies that all the wave in the list exist, and convert them all to 8 kHz so they can be played over the phone.
Save wave list - To a file. You can include this file as a resource in your application, which you can load and pass to ISTTTSQueue::WaveAddFromList List or ITellnfo::QuickCreate.

Figura 2. Con el editor de listas crearemos nuestros propios enlaces a archivos de sonido.

```
  phone1.Speak lineID, "Los siguientes seis
      dígitos de su tarjeta de crédito son "
  phone1.Speak lineID, s
  CNextSix = s
'verify!
  phone1.YesNoFromString lineID, "[Prompts]"
      + vbNewLine + _
     "Main=es correcto?" + vbNewLine + _
     "where=obteniendo los 6 números
      siguientes" + vbNewLine + _ "[Set-
      tings]" + vbNewLine + _
     "CanGoBack=1" + vbNewLine, result, s
  If (result < 0) Then
   GoTo done
  End If
  If (result <> 1) Then
   GoTo nextsixcredit
  End If
```

Ahora obtenemos los últimos 5 números de la tarjeta de crédito:

```
lastfivecredit:

s = GetNumber(lineID, 5, " Por favor, introduz-
      ca los siguientes 5 dígitos del número de
      su tarjeta de crédito ", _"qué son los últi-
      mos 5 dígitos?", _"está introduciendo los
      últimos 5 dígitos de su tarjeta de crédito",
      _"Dígame el número de su tarjeta de cré-
      dito.")

  If (s = "") Then
   GoTo done
  End If
    phone1.Speak lineID, "los últimos 5 dígitos
      de su tarjeta de crédito son "
  phone1.Speak lineID, s
  CLastFive = s
```

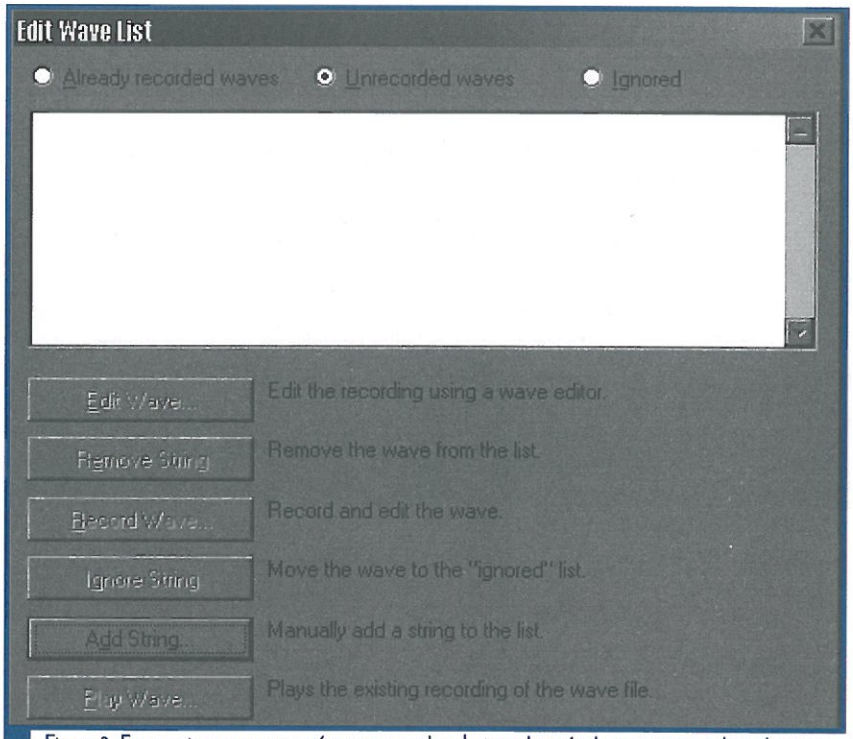


Figura 3. En esta imagen vemos cómo se procede a la introducción de una nueva cadena de texto en el editor de listas de WAVs.

Una vez más, procederemos a verificar que los datos que han sido introducidos por el usuario sean correctos:

```
  phone1.YesNoFromString lineID, "[Prompts]"
      + vbNewLine + _
     "Main=es correcto?" + vbNewLine + _
     "where=obteniendo los últimos 5 núme-
      ros" + vbNewLine + _
     "[Settings]" + vbNewLine + _
     "CanGoBack=1" + vbNewLine, result, s
  If (result < 0) Then
   GoTo done
  End If
  If (result <> 1) Then
   GoTo lastfivecredit
  End If

  phone1.Speak lineID, "ahora necesitamos la
      fecha de caducidad de la tarjeta. Por favor,
      diga mes y año, por ejemplo, si la fecha es
      marzo de 1999 diga 3 barra 99, o en el
      teléfono presione 0 3 9 9"

expiration:
```

Puesto que el formato de la fecha de una tarjeta de crédito es único utilizaremos un formato a partir de un fichero **date.txt**. Para ello se empleará una función denominada *GrammarFromFile*:

```
  phone1.GrammarFromFile lineID, App.Path +
      "\date.txt", result, s
    If (s = "") Then
   GoTo done
  End If

  If (result < 0) Then
```

Tabla 1. Método Speak.

Speak(lineID As Long, text As String)

Utiliza el texto asignado en text para reproducirlo. Si el parámetro text corresponde a una de las cadenas de caracteres utilizadas por las funciones WaveAddFromListFile, WaveAddFromListResource, WaveAddFromListString o WaveAddFromMemory, se reproducirá el archivo WAV correspondiente en lugar de hablar el texto asignado a la variable.

Parámetro	Descripción
lineID	[in] Identificación de la línea de teléfono. La aplicación ha pasado este número cuando recibe un evento Initialize o DoPhoneCall.
text	[in] Texto a hablar.

Tabla 2. Método YesNoFromString.

YesNoFromString(lineID As Long, Grammar As String, result As Long, parse As String)

Carga un control Yes/No a partir de un string y activa el control de telefonía hablando y reconociendo.

Parámetro	Descripción
lineID	[in] Identificación de la línea de teléfono. La aplicación pasa este número cuando se ejecuta el evento Initialize o DoPhoneCall.
Grammar	[in] Cadena de texto.
result	[out] Resultado: 1 para yes o 2 para no
parse	[out] Actualmente no utilizada.

```
GoTo done
End If
```

Llamaremos a *NormalizeDate* para asegurarnos de que la salida está en el formato *# espacio # barra # espacio #*. La barra la hemos agregado en el programa a propósito para diferenciar las partes de la fecha que se han introducido (meses, años, etc.)

```
CExpDate = NormalizeDate(s)

phone1.Speak lineID, "La fecha de caducidad
    de la tarjeta es "
phone1.Speak lineID, CExpDate
```

Validamos la fecha obtenida:

```
phone1.YesNoFromString lineID, "[Prompts]"
    + vbNewLine + _
    "Main=es correcto?" + vbNewLine + _
    "where=obtenemos fecha de caducidad" +
     vbNewLine + _
    "[Settings]" + vbNewLine + _
    "CanGoBack=1" + vbNewLine, result, s

If (result < 0) Then
    GoTo done

End If
 If (result <> 1) Then
  GoTo expiration
 End If
```

El siguiente paso consiste en obtener el titular de la tarjeta. Por supuesto, existen demasiados nombres y pronunciaciones de nombres para reconocer, por lo que lo grabaremos directamente al disco duro. Si necesitáramos deletrear exactamente el nombre para verificaciones, tendríamos que codificar esa información en la clave generada por el programa.

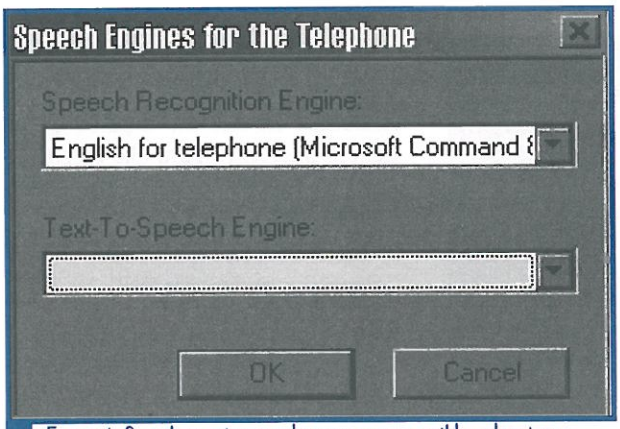


Figura 4. Cuando se ejecuta el programa es posible seleccionar cualquiera de los engines instalados en el equipo.

```
name:
  Dim wave As Long
  Dim wsize As Long

  phone1.Record-
FromString lineID,
"[Prompts]" + vbNew-
Line + _
    "Main=Qué es el titular?" + vbNewLine +
     _
    "Main.2=Qué es el nombre de la tarjeta" +
     vbNewLine + _
    "Where=Obteniendo nombre de la tarjeta"
     + vbNewLine + _
    "Help=Busco el nombre" + vbNewLine,
     result, wave, wsize

If (result < 0) Then
    GoTo done
  End If
```

TTS reproducirá un *WAV* si colocamos una palabra entre <> y utilizará *WaveAddFromMemory* para indicar el fichero.

```
phone1.WaveAddFromMemory lineID,
    "<nombre>", wave, wsize
phone1.Speak lineID, "el nombre es"
phone1.Speak lineID, "<nombre>"
```

A continuación podemos pasar a verificar el nombre introducido del siguiente modo:

```
phone1.YesNoFromString lineID, "[Prompts]"
    + vbNewLine + _
   "Main=es correcto?" + vbNewLine + _
   "where=verificando nombre" + vbNewLi-
    ne + _
   "[Settings]" + vbNewLine + _
   "CanGoBack=1" + vbNewLine, result, s
If (result < 0) Then
     phone1.FreeWave wave
     GoTo done
  End If

If (result <> 1) Then
 phone1.FreeWave wave
 GoTo name
End If
```

Para finalizar verificamos todos los datos obtenidos:

```
phone1.Speak lineID, "Ahora verifiquemos la
     información que usted me ha proporciona-
     do. Yo tengo la clave como " + sPassKey
     + ", la tarjeta de crédito como " +
     CFirstFour + CNextSix + CLastFive +
     "la fecha de caducidad como " + CExp-
     Date + ", y el nombre como "
```

```
phone1.Speak lineID, "<nombre>"

phone1.YesNoFromString lineID, "[Prompts]"
    + vbNewLine + "Main=es correcto?" +
    vbNewLine + "where=verificando todos
    los datos" + vbNewLine + "[Settings]"
    + vbNewLine + "CanGoBack=1" +
    vbNewLine, result, s

If (result <> 1) Then
 phone1.FreeWave wave
 GoTo PassKey
End If
```

'Liberamos la reproducción del archivo *WAV*.

```
phone1.FreeWave wave
```

Preguntamos si todo es correcto para hacer el cargo a la tarjeta:

```
phone1.YesNoFromString lineID, "[Prompts]"
    + vbNewLine + "Main=el cargo serán
    mil pesetas. De acuerdo??" + vbNewLine
    + "where=verificando precio" + vbNew-
    Line + "[Settings]" + vbNewLine +
    "CanGoBack=1" + vbNewLine, result, s

If (result < 0) Then
    GoTo done
  End If

If (result <> 1) Then
 GoTo NoPay
End If
```

La llave de registro puede ser encriptada y guardada en el registro de Windows

A continuación generaríamos una clave de registro real, basada en la clave que introdujo el usuario. Pero como esto es un ejemplo, daremos por válido un número determinado de registro (**CSB666**):

```
phone1.Speak lineID, "Su clave de registro es C
    S B 6 6 6. Introdúzcalo en el cuadro de
    diálogo de registro de su programa para
    activar todas las características del pro-
    ducto Shareware. Repito, su clave de
    registro es C S B 6 6 6. Gracias por com-
    prar uno de nuestros productos"

repeat:

  phone1.YesNoFromString lineID, "[Prompts]"
    + vbNewLine + "Main=La clave es  C S
    B 6 6 6. La ha copiado ya?" + vbNewLine
    + "where=Dando la tarjeta de registro "
    + vbNewLine + "[Settings]" + vbNew-
    Line + "CanGoBack=1" + vbNewLine,
    result, s

  If (result < 0) Then
      GoTo done
    End If

  If (result <> 1) Then
   GoTo repeat
  End If

  GoTo done
```

En caso de que el cliente no quiera pagar...

```
NoPay:

phone1.Speak lineID, "Lo siento, pero debe
    estar de acuerdo en el pago de las mil
    pesetas para obtener la versión completa
    del producto. Gracias por llamar. Adiós."

  GoTo Done2
done:
  phone1.Speak lineID, "Adiós"
Done2:
End Sub
```

Un usuario debe registrarse para obtener la versión completa

La siguiente función de ayuda nos asegura que la fecha siempre tenga el mismo formato:

```
Private Function NormalizeDate(thedate As
    String) As String
Dim inputd As String
Dim ds As String
Dim c As String

inputd = thedate      'Duplicamos el string
ds = ""               'Comienza con un
    valor de retorno vacío
slen = Len(inputd)
```

Extrae todos los caracteres que no sean números y agrega un espacio entre cada número:

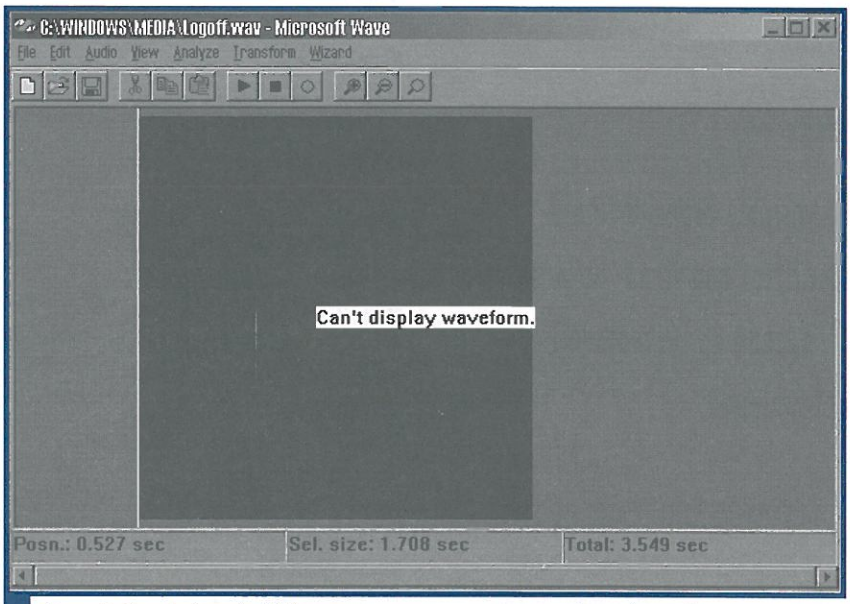

C:\WINDOWS\MEDIA\Logoff.wav - Microsoft Wave
File Edit Audio View Analyze Transform Wizard
Can't display waveform.
Posn.: 0.527 sec
Sel. size: 1.708 sec
Total: 3.549 sec

Figura 5. Con el editor de WAVs crearemos nuestros propios archivos de sonido.

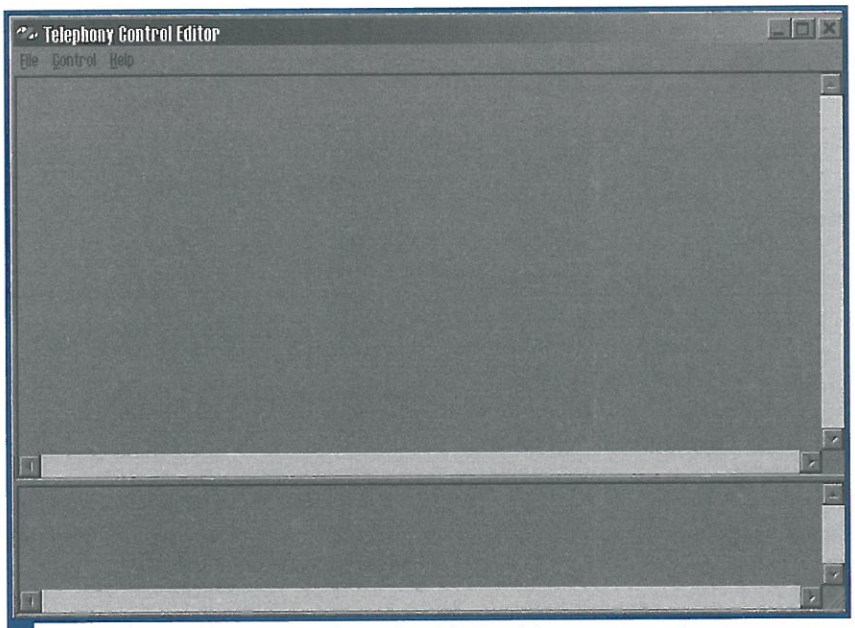


Figura 6. La API Speech posee un editor de controles de telefonía para generarlos a nuestro gusto.

```
For i = 1 To slen
   c = Mid$(inputd, i, 1)
   If (c >= "0") And (c <= "9") Then
      ds = ds + c + " "
   End If
   If (i = 2) Then
```

'Agregamos una barra entre los números de los meses y años

```
      ds = ds + "barra "
   End If
Next i

NormalizeDate = ds
End Function
```

La variable especial lineID identifica qué línea/tarea se está utilizando en un momento determinado

Función de ayuda para obtener un número:

```
Private Function GetNumber(lineID As Long,
      numdigits As Integer, Main As String,
      main2 As String, where As String, help
      As String) As String

Dim result As Long
Dim s As String
Dim tstr As String

Dim NumGram As String
```

Establecemos un rango de 0 a 9 para obtener todos los números que sean posibles:

```
NumGram = ""
For i = 1 To numdigits
   NumGram = NumGram + "<0..9> "" """
Next i

phone1.GrammarFromString lineID,
      "[Prompts]" + vbNewLine + _
   "Main=" + Main + vbNewLine + _
"Main.2=" + main2 + vbNewLine + _
"Where=" + where + vbNewLine + _
"Help=" + help + vbNewLine + _
"[DTMFString]" + vbNewLine + _
"Count=" + Str(numdigits) + vbNewLine + _
"0=0" + vbNewLine + _
"1=1" + vbNewLine + _
"2=2" + vbNewLine + _
"3=3" + vbNewLine + _
"4=4" + vbNewLine + _
"5=5" + vbNewLine + _
"6=6" + vbNewLine + _
"7=7" + vbNewLine + _
"8=8" + vbNewLine + _
"9=9" + vbNewLine + _
"[<MyGrammar>]" + vbNewLine + _
"<MyGrammar>= " + NumGram + vbNewLi-
      ne, result, s
```

Preparamos **GetNumber** como **Null** ("") para empezar a obtener números de la variable **tstr:**

```
GetNumber = ""
tstr = s
For i = 1 To Len(tstr)
   GetNumber = GetNumber + Mid(tstr, i, 1)
      + " "
Next i

End Function
```

CONCLUSIÓN

Hemos acabado una serie de artículos en los cuales se han detallado los aspectos más importantes de la programación del reconocimiento de voz y sus aplicaciones en la vida real. Pero por supuesto, esto no acaba aquí, puesto que ahora es cuando el programador debe poner sobre el teclado todas sus cualidades y conocimientos para emplear lo aprendido de la mejor forma.

Queda a expensas del programador realizar el código necesario para validar y cargar dicha tarjeta

Como se ha comentado en diversos números de esta serie, el reconocimiento de voz aún es susceptible de ser mejorado, pero se acerca a la máxima de que el ordenador pueda comprender lo que queremos expresarle de una forma más humana...

www.dadvina.com

Demo disponible en nuestra Web.

Algunos Ratones son muy curiosos...

¡Proteja la información de sus aplicaciones!

DadvinaFence
Seguridad en sus Aplicaciones.

Uno de los riesgos que sufren las aplicaciones informáticas es la posibilidad de su uso por personas no autorizadas.
DadvinaFence incorpora tres componentes OCX o VCL:**ApplicationLock, FormLock** y **NetMessenger**, proporcionando al programador un sistema de seguridad activo fácil de integrar y totalmente configurable.

- Versiones VCL para Delphi 3.0/2.0, C++ Builder 3 y OCX para Visual Basic 5.0.
- Asignación de permisos y privilegios sobre los objetos existentes en las pantallas de las aplicaciones.
- Autentificación de usuarios mediante el uso de identificación y contraseña.
- Detección de intentos fallidos continuados de entrada al sistema,avisando al instante a los administradores de seguridad de las aplicaciones.
- Centralización de los accesos (login) a las bases de datos de la aplicación.
- Registro de entradas y salidas.
- Sistema modular fácilmente actualizable en tiempo de ejecución y de diseño.

etc...

Sus clientes se lo Agradecerán

Sólo por 31.500* ptas.

Dadvina Systems.

San Toribio 6.
28031 Madrid.
Email: com@dadvina.com

Para más información o realizar pedidos contacte con el telefono 91 380 32 46 o en nuestra Web: www.dadvina.com.

* IVA no incluido.

HTML Dinámico (IV). Menús desplegables

Adolfo Aladro (aaladro@arrakis.es)

Y son cuatro... A lo largo de este artículo vamos a continuar tratando de exprimir, al máximo, todo el jugo a HTML Dinámico. Esta vez, y como complemento a los tan importantes conceptos ya aprendidos anteriormente, vamos a hacerlo a través de un nuevo y útil efecto: los menús desplegables.

MENU

Se trata de crear una serie de opciones en la zona superior de la página de forma que al pasar el ratón por encima de cualquiera de ellas se muestre un menú. Haciendo clic sobre cualquiera de las opciones iremos a una nueva página que a su vez contendrá del mismo modo un menú. De esta manera iremos creando un índice organizado y elegante mediante el cual podemos navegar por todas las páginas de un sitio Web cómodamente (Ver Figura 1). Tal y como hemos venido haciendo, hasta este momento volveremos a utilizar nuestro ya famoso y conocido por todos script "dhtml.js" y al mismo tiempo procederemos a introducir los nuevos e interesantes conceptos que vendrán a complementar los ya aprendidos anteriormente.

UN REPASO AL DOM (DOCUMENT OBJECT MODEL)

Aquellos lectores que hayan seguido la serie de artículos sobre el HTML Dinámico compatible seguramente ya tendrán a estas alturas muy claro cuales son las diferencias principales entre NS4 y IE4 a la hora de acceder a los distintos objetos. Sin embargo, a propósito de este nuevo capítulo, no viene nada mal que volvamos sobre este tema ya que en el ejemplo que vamos a implementar surgirán nuevas situaciones que hasta ahora no habíamos visto.

Cuando nosotros creamos una capa dentro de una página HTML utilizando la etiqueta DIV

```
<DIV ID="micapa" CLASS="mestilo">
    ...
</DIV>
```

podemos acceder a ella dependiendo del navegador de la siguiente forma:

```
IE4   document.all["micapa"]
NS4   document["micapa"]
```

Estas referencias hacen posible que podamos manejar las capas. Pero si lo que queremos realmente es acceder a las propiedades de las capas relacionadas con su estilo, ya sean propiedades de posicionamiento (*top*, *bottom*, etc.) o de presentación (*color*, *font*, etc.), entonces debemos tener en cuenta que en IE4 la referencia varía debiendo ser:

```
IE4   document.all["micapa"].style
```

¿Por qué ponemos ahora de manifiesto esta diferencia? La razón es que

Listado 1. Definición de los arrays que contienen los elementos de los menús.

```
var menus = new Array();

menus[0] = new Array("C/C++", "C/C++(1)", "cpp01.htm", "C/C++(2)", "cpp02.htm");

menus[1] = new Array("Java", "Programación Java(1)", "java01.htm", "Programación
Java(2)", "java02.htm", "Recursos Java(1)", "java03.htm", "Recursos Java(2)",
"java04.htm", "Recursos Java(3)", "java05.htm");

menus[2] = new Array("Perl", "Programación Perl(1)", "perl01.htm", "Programación
Perl(2)", "perl02.htm", "Tutorial Perl", "perl03.htm");

menus[3] = new Array("XML", "Programación XML", "xml01.htm", "Tutorial XML",
"xml02.htm", "Enlaces XML", "xml03.htm");
```

en el ejemplo que nos ocupa vamos a asignar manejadores de eventos a capas. Es decir, dentro de nuestro script dotaremos a las capas de la capacidad de detectar eventos tales *onmouseover* o *onclick*. Al hacer esta asignación tendremos que prestar especial cuidado de hacerlo con respecto a las capas. Esto en el caso de NS4 no reviste mayor dificultad ya que como hemos visto la referencia es la misma tanto para las propiedades del estilo de la capa como para las demás. Pero en IE4 no es así. En resumen, la referencia global que creamos en el script "dhtml.js" es una referencia al estilo de la capa y en el caso de IE4 no nos servirá para asignar eventos.

Por ejemplo, si creamos una capa de la siguiente manera:

```
c_micapa = new objetoCapa("micapa");
```

y posteriormente asignamos un evento:

```
c_micapa.onmouseover = hacerMouseOver;
```

Esto sólo tendrá validez en el caso de tratarse de NS4 ya que internamente es como si estuviéramos haciendo:

```
IE4  document.all["micapa"].style.onmouseo-
     ver = hacerMouseOver;
NS4  document["micapa"].onmouseover =
     hacerMouseOver;
```

Lo que en IE4 es evidentemente un error ya que tratamos de asignar un evento al estilo de la capa y no a la capa en sí. Por todo ello en nuestro script vamos a tener que trabajar en numerosas ocasiones de nuevo con la función *eval* y las variables globales *layerObj* y *styleObj*.

Listado 2. Función que se encarga de crear dinámicamente las capas que forman el menú.

```
function menuPrincipal() {
 codHTML = '';
 for(i=0;i<menus.length;i++) {
  codHTML += '<DIV ID="menup_' + i + '" CLASS="mestilo">';
  codHTML += menus[i][0];
  codHTML += '</DIV>';
  numero_submenus = (menus[i].length-1)/2;
  for(j=0;j<numero_submenus;j++) {
   codHTML += '<DIV ID="menup_' + i + '_' + j + '" CLASS="miestilo">';
   k = 2*j + 1;
   codHTML += menus[i][k];
   codHTML += '</DIV>';
  }
 }
 return codHTML;
}
```

EL ARRAY DE LOS ELEMENTOS DE LOS MENÚS Y LA CREACIÓN DINÁMICA DE LAS CAPAS

Siempre que se crea un script es conveniente meditar la posibilidad de que ese script pueda servir en el caso de que algunos parámetros cambien. Para nuestro ejemplo esto es muy importante. Tenemos que crear los menús de forma tal que si en un futuro aumentan o disminuyen sus opciones el script se adapte automáticamente a las nuevas circunstancias. Por ello definimos una serie de arrays que contienen las opciones de los menús y el script simplemente crea tantos menús como los arrays indiquen, posicionándolos además correctamente en la página HTML. Si queremos cambiar las menús tan solo tendremos que modificar los arrays y no será necesario tocar el script. El Listado 1 muestra como son estos arrays en nuestro ejemplo.

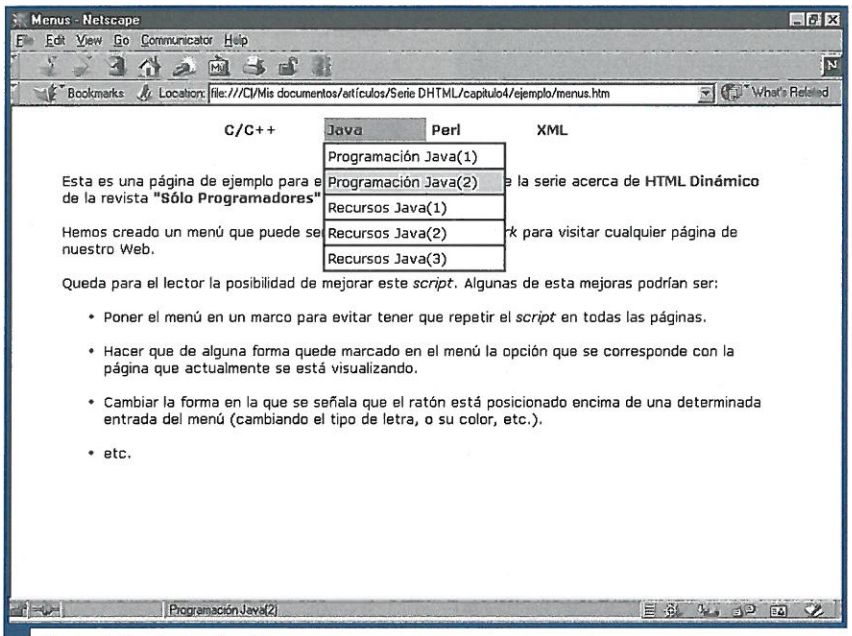

Menus - Netscape
File Edit View Go Communicator Help
Bookmarks Location: file:///C|/Mis documentos/artículos/Serie DHTML/capitulo4/ejemplo/menus.htm
What's Related
C/C++
Java
Perl
XML
Programación Java(1)
Programación Java(2)
Recursos Java(1)
Recursos Java(2)
Recursos Java(3)
Esta es una página de ejemplo para e
e la serie acerca de HTML Dinámico
de la revista "Sólo Programadores"
Hemos creado un menú que puede se
rk para visitar cualquier página de
nuestro Web.
Queda para el lector la posibilidad de mejorar este script. Algunas de esta mejoras podrían ser:
• Poner el menú en un marco para evitar tener que repetir el script en todas las páginas.
• Hacer que de alguna forma quede marcado en el menú la opción que se corresponde con la página que actualmente se está visualizando.
• Cambiar la forma en la que se señala que el ratón está posicionado encima de una determinada entrada del menú (cambiando el tipo de letra, o su color, etc.).
• etc.
Programación Java(2)

Figura 1. Menús desplegables.

El menú tendrá tantas entradas como elementos tenga el array *menu*. En nuestro caso serán cuatro: "C/C++" (*menu[0]*), "Java" (*menu[1]*), Perl (*menu[2]*) y "XML" (*menu[3]*). Cada elemento del array menú será a su vez otro array donde la primera posición estará ocupada por el nombre de la entrada y las siguiente se organizarán sucesivamente de la siguiente forma: nombre de la opción, página a la que enlaza, nombre de la opción, página a la que enlaza, etc. (Ver la Figura 2).

Cada elemento que forma parte del menú, ya sean las entradas principales o cada una de las opciones, es una capa. El número de capas y sus contenidos dependerán finalmente de los arrays definidos. La función *menuPrincipal* se encarga de generar el código HTML correspondiente a las capas y esta función es llamada dentro de cada página. El Listado 2 muestra la función *menuPrincipal*. Como se puede observar hay dos bucles: uno que recorre el array *menu* y otro que recorre los arrays *menu[i]*. Es importante fijarse en cómo vamos a llamar a las capas. Las entradas de los menús tendrán los nombres *menup_0*, *menup_1*, *menup_2* y así sucesivamente (tantas como elementos contenga el array *menu*). Las opciones que están asociadas a cada entrada se llamarán *menup_0_0*, *menup_0_1*, *menup_0_2* y así sucesivamente para la entrada del menú *menup_0*; *menup_1_0*, *menup_1_1*, *menup_1_2* y así sucesivamente para la entrada del menú *menup_1*; etc.

El Listado 3 nos muestra un ejemplo de una página HTML en la que incluimos nuestro menú. Lo primero que se hace nada más empezar a cargar el cuerpo de la página es escribir todo el código HTML correspondiente a las capas, código este que es proporcionado por la función *menuPrincipal*.

Listado 3. Ejemplo de página que contiene el menú.

```
<HTML>
<HEAD>
<TITLE>Java</TITLE>
<SCRIPT LANGUAGE="JavaScript" SRC="dhtml.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript" SRC="menus.js"></SCRIPT>
</HEAD>
<BODY onload="iniciar()">
<SCRIPT LANGUAGE="JavaScript">
<!—
 document.write(menuPrincipal());
//—>
</SCRIPT>
<BR> <BR>
<CENTER><BIG>Recursos Java(1)</BIG></CENTER>
<BR> <BR>
<CENTER><A HREF="menus.htm">Volver al principal</A></CENTER>
</BODY>
</HTML>
```

LA FUNCIÓN INICIAR

Como en casi todos los scripts que hemos desarrollado existe una función llamada iniciar que se invoca justo en el momento que se produce el evento *onload* de la página HTML. Esta función se encarga de inicializar todos los parámetros correctamente de forma que el menú se visualice correctamente y además esté preparado para interaccionar con el usuario.

A grandes rasgos podemos decir que la función iniciar se compone, al igual que *menuPrincipal*, de dos bucles que recorren respectivamente el array

menu y cada uno de los arrays *menu[i]*. Con cada una de las capas tenemos que realizar básicamente dos tareas: posicionarla correctamente para que el menú tome forma y de la impresión de ser un todo, no un conjunto de capas dispersas; y asignar los eventos que se correspondan en cada caso.

POSICIONAMIENTO DE LAS CAPAS

En primer lugar tenemos que asegurarnos de que las capas asociadas a cada entrada del menú se posicionan correctamente. Para dotar al script de mayor flexibilidad hemos creado las variables globales *m_posY* y *m_ancho_entrada*. La primera expresa la posición vertical del menú dentro de la página HTML que lo contiene, y la segunda contiene el valor del ancho de cada entrada del menú. El Listado 4 muestra cómo posicionamos verticalmente estas capas y cómo calculamos la posición horizontal para cada una de ellas. Nótese que para que el menú quede centrado horizontalmente primero hemos tenido que inicializar la variable *posX* de la siguiente forma:

Listado 4. Código que determina la posición de las capas asociadas a las entradas del menú.

```
eval('c_menup_' + i + '.left = posX');
eval('c_menup_' + i + '.top = m_posY');
  eval('posX = parseInt(c_menup_' + i + '.left) + c_menup_' + i + '.getWidth();');
  eval('posY = parseInt(c_menup_' + i + '.top) + c_menup_' + i + '.getHeight();');
```

```
posX = (ancho_ventana -
      m_ancho_entrada*menus.length)/2;
```

La variable posY que aparece en el Listado 4 nos sirve para posicionar verticalmente la primer de las opciones vinculadas con esa entrada del menú. La manera en la que vamos colocando las capas de las opciones es similar a la que hemos utilizado para posicionar las capas de las entradas. El Listado 5 muestra cómo lo hacemos.

Dependiendo de cómo hayamos definido los estilos de las capas (tamaño del borde, valor de *padding*, etc.) habrá que reajustar convenientemente la manera de colocar todas capas, pero en líneas generales el método será siempre el mismo.

Conviene no olvidar nunca la utilización de la función *parseInt* cuando para hacer un cálculo partimos de las propiedades *top* o *left* de alguna capa. Esto es necesario ya que IE4 no almacena valores numéricos en esas propiedades sino cadenas del tipo "200px" con lo que si omitimos la utilización de la función parseInt se producirá un error.

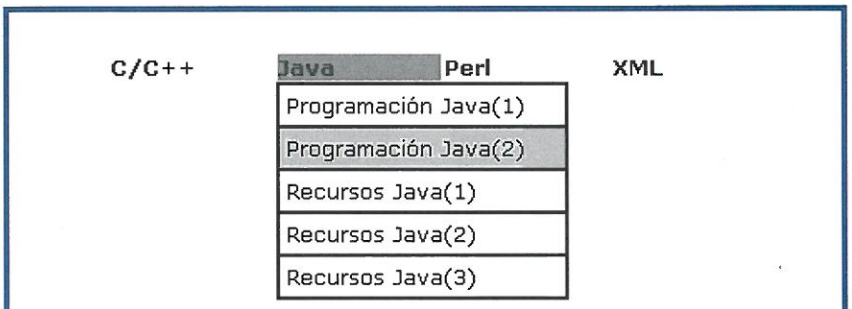


Figura 2. Correspondencia entre el array y el menú.

ASIGNACIÓN DE EVENTOS

La segunda tarea más importante que tenemos que llevar a cabo dentro de la función iniciar es asignar los eventos adecuados a cada capa. En las páginas HTML la forma más usual de trabajar con eventos es insertando en las propias etiquetas los manejadores de eventos. Así podemos hacer por ejemplo que cuando el ratón pase por encima de un enlace ocurra algo determinado. Basta con incluir asociar un manejador (en otras palabras, una función JavaScript) en la etiqueta HTML correspondiente.

```
<A HREF="mipagina.htm"
      onmouseover="hacerMouseOver()">Est
      o es un enlace</A>
```

Con la aparición de NS4 y IE4 la gestión de los eventos se vio considerablemente mejorada y ahora es posible, entre otras muchas cosas, asignar eventos directamente a objetos mediante JavaScript.

En el ejemplo que nos ocupa tendremos que detectar el evento *onmouseover* para las capas que contienen las entradas y los eventos *onmouseover*, *onmouseout* y *onclick* para aquellas que contienen las opciones de cada entrada. El Listado 6 contiene el código

Listado 5. Código que determina la posición de las capas asociadas a las opciones de cada entrada del menú.

```
eval('c_menup_' + i + '_' + j + '.top = posY');
eval('c_menup_' + i + '_' + j + '.left = parseInt(c_menup_' + i + '.left,10);');
eval('posY = parseInt(c_menup_' + i + '_' + j + '.top) + c_menup_' + i + '_' + j +
'.getHeight() - 2;');
```

Listado 6. Asignación de eventos a las capas que contienen las entradas del menú.

```
if (NS4) {
  eval('c_menup_' + i + '.onmouseover = hacerMenuPrincipalMouseOver;');
  eval('c_menup_' + i + '.n_menu = i;');
} else {
  eval(layerObj + '["menup_' + i + '"].onmouseover = hacerMenuPrincipalMouseO-
ver;');
  eval(layerObj + '["menup_' + i + '"].n_menu = i;');
}
```

JavaScript que asigna los manejadores correspondientes a las capas que contienen las entradas del menú. Como ya hemos explicado al comienzo de este artículo es preciso hacerlo de una forma diferente dependiendo de si se trata de NS4 o IE4. Además de asignar el manejador también creamos una nueva propiedad de la capa llamada *n_menu* que contiene el número del menú correspondiente. La utilidad de esta propiedad se irá viendo más adelante cuando vayamos analizando las funciones manejador. El Listado 7 contiene el código JavaScript que va a realizar la asignación de manejadores para las capas que contienen las opciones de cada entrada del menú. De la misma forma que en el caso anterior aquí también crearemos dos propiedades nuevas para cada uno: *enlace* y *texto*. La primera de ellas contendrá la dirección URL de la página a la que enlaza esa opción y la segunda el texto que aparecerá cuando se muestra el menú.

Listado 7. Asignación de eventos a las capas que contienen las opciones de cada entrada del menú.

```
if (NS4) {
  eval('c_menup_' + i + '_' + j + '.onmouseover = hacerMouseOver;');
  eval('c_menup_' + i + '_' + j + '.onmouseout = hacerMouseOut;');
  eval('c_menup_' + i + '_' + j + '.captureEvents(Event.MOUSEUP);');
  eval('c_menup_' + i + '_' + j + '.onmouseup = hacerClick;');
  eval('c_menup_' + i + '_' + j + '.enlace = menus[i][k+1]');
  eval('c_menup_' + i + '_' + j + '.texto = menus[i][k]');
} else {
  eval(layerObj + '["menup_' + i + '_' + j + '"].onmouseover = hacerMouseOver;');
  eval(layerObj + '["menup_' + i + '_' + j + '"].onmouseout = hacerMouseOut;');
  eval(layerObj + '["menup_' + i + '_' + j + '"].onfocus = hacerClick;');
  eval(layerObj + '["menup_' + i + '_' + j + '"].enlace = menus[i][k+1];');
eval(layerObj + '["menup_' + i + '_' + j + '"].texto = menus[i][k];');
}
```

LOS MANEJADORES DE EVENTOS

Las funciones que manejan los distintos eventos que se producen relacionados con el menú son muy importantes. Cada vez que el ratón pasa por encima de una de las entradas del menú hay que mostrar las opciones correspondientes a esa entrada y ocultar las opciones de otra entrada en el caso de que anteriormente ya estuviera mostrándose otra entrada. Además tenemos que tener en cuenta que el usuario no tiene siempre porque hacer clic en una de las opciones cada vez que navegue por el menú. Es decir, tenemos que prever un mecanismo que oculte las opciones pasado un tiempo. Este espacio de tiempo será de compromiso: un lapso lo suficientemente grande como para que al usuario le de tiempo a buscar por el menú la opción que desea, y lo suficientemente corto como para que no estén desplegadas las opciones demasiado tiempo.

El Listado 8 muestra la función manejador asociado a los eventos *onmouseover* de las entradas del menú. Cada vez que usuario hace pasar el ratón por encima de una entrada del menú se cancela el temporizador que estuviera activo y se lanza de nuevo. El resto de las tareas son obvias: ocultamos las opciones que anteriormente se estuvieran mostrando (si es que las había); marcamos esta entrada como la última que se ha visitado (esta variable servirá precisamente para que podamos posteriormente ocultar estas opciones cuando visitemos una nueva entrada); y se muestran las opciones de esta entrada.

El Listado 9 muestra los manejadores de los eventos *onmouseover* y *onmouseout* de las opciones de cada entrada del menú. Estas son realmente sencillas ya que lo único que hacen es

Listado 9. Manejadores de los eventos onmouseover y onmouseout que contienen las opciones de cada entrada del menú.

```
function hacerMouseOver(e) {
 if (NS4) {
  this.bgColor = mi_color_dentro;
 } else {
  this.style.background = mi_color_dentro;
 }
 self.status = this.texto;
 return true;
}

function hacerMouseOut(e) {
 if (NS4) {
  this.bgColor = mi_color_fuera;
 } else {
  this.style.background = mi_color_fuera;
 }
}
```

cambiar el color de fondo de la capa y actualizar la barra de estado del navegador adecuadamente.

En todas estas funciones estamos viendo la gran utilidad de añadir propiedades a las capas. De esta forma tenemos un acceso rápido y elegante a las propiedades de las capas y somos capaces de saber rápidamente cuál es la dirección URL de cada opción, el texto de misma o el número de esa entrada.

Listado 8. Manejador del evento onmouseover de las capas contienen las entradas del menú.

```
function hacerMenuPrincipalMouseOver(e) {
 clearTimeout(temporizador);
 if (m_mostrandose != -1) {
  ocultarTodo(m_mostrandose);
  if (NS4) {
   eval('c_menup_' + m_mostrandose + '.bgColor = m_color_fuera');
  } else {
   eval('c_menup_' + m_mostrandose + '.background = m_color_fuera');
  }
 }
 if (NS4) {
  this.bgColor = m_color_dentro;
 } else {
  this.style.background = m_color_dentro;
 }
 m_mostrandose = this.n_menu;
 numero_submenus = (menus[this.n_menu].length-1)/2;
 for(i=0;i<numero_submenus;i++) {
  eval('c_menup_' + this.n_menu + '_' + i + '.mostrar()');
 }
 temporizador = setTimeout('hacerDocumentClick()', 3000);
}
```

Lo fundamental es comprender el mecanismo de interacción entre unos eventos y otros. Una vez que hayamos hecho esto basta con echar un vistazo a las distintas funciones para saber como funciona todo el sistema.

EL TRUCO DEL COLOR DE FONDO

Hasta ahora hemos pasado por alto las explicaciones relativas a las propiedades de los estilos de los distintos elementos del menú que tienen que ver con su presentación: color de fondo, letra, borde, etc. En este punto lo que más importa es que se vayan probando con distintos valores a fin de lograr un menú que nos satisfaga visualmente pero también es preciso conocer algunos problemas que se derivan de las diferencias existentes entre NS4 y IE4 a la hora de implementar los estilos.

En primer lugar tenemos el problema del color de fondo. Cuando definimos el estilo de una capa y queremos asignar un color de fondo que posteriormente pueda ser modificado debemos hacerlo de manera distinta dependiendo del navegador: con IE4 utilizaremos la propiedad *background* mientras que con NS4 utilizaremos la propiedad *layer-background-color*. Esto será en la definición de los estilos, luego en el script accederemos a la propiedad *bgColor* de la capa en el caso de tratarse de NS4 y a la propiedad *background* del estilo de la capa para IE4.

Otro inconveniente importante es la forma en la que NS4 interpreta las propiedades *padding* y *border* de los estilos. Aquí NS4 presenta numerosas anomalías que no se producen en IE4 así que tendremos que probar con distintas combinaciones de esas propiedades, diferentes para cada navegador, hasta que logremos un efecto similar para los dos casos.

Creación de un buscador Web (III)

Enrique de la Lastra (elastra@redestb.es)

Un robot implementado como soporte de un buscador Web debe poder analizar cada página HTML que se le pase como parámetro, guardar una referencia de la misma y de igual modo recuperar y volver a analizar todas las páginas HTML enlazadas.

INTRODUCCIÓN

En los dos artículos anteriores de la serie sobre *Creación de un Buscador Web* se introdujeron los principales conceptos necesarios para la tarea de desarrollar un *Robot Web*, el cual es, a la postre, el corazón de un Buscador. En el presente artículo, habiendo sentado ya las bases de nuestro trabajo, comenzaremos a desarrollar nuestro propio *robot* de búsqueda y análisis de páginas *Web*. Por las limitaciones de espacio inherentes al propio artículo, únicamente implementaremos la parte del código que se va a encarga de realizar las peticiones *HTTP* a cada servidor *Web* que pasemos como parámetro. Más concretamente, la petición se realizará a la *URL* seleccionada por el usuario de la aplicación, la cual, lógicamente se podrá modificar.

PANTALLA PRINCIPAL

En la Figura 1 se muestra el aspecto de la pantalla principal de nuestro *robot Web* - con el resultado obtenido tras analizar la página principal del servidor *Web* de *Tower* (http://www.towercom.es). Como se puede observar, hemos definido un cuadro de texto donde el usuario puede teclear la *URL* que quiera analizar (recordando que si no se introduce el nombre de la página *HTML*, el servidor *Web* devolverá la página por defecto que haya prefijado) y un botón (*Analizar*) que arrancará el proceso de dicha *URL*.

También se ha definido un campo de tipo *Memo* para mostrar el estado de la conexión Cliente-Servidor, y por último, un conjunto de cuadros de texto y "memos" en los que mostraremos las partes que nos interesan de cada página *HTML* analizada. En concreto: las etiquetas *META* para *ROBOTS*; el título (<*TITLE*>) de la página; el primer título destacado del cuerpo del documento (<*H1*> ó <*H2*>); el primer párrafo de texto dentro del cuerpo y una lista con los enlaces a otras páginas *Web*. El botón *Analizar siguiente* será encargado de analizar la página que el usuario seleccione de la lista. Por el momento, este botón estará inhabilitado ya que en un principio la lista de enlaces mostrada sólo se utiliza a modo informativo.

LIMITACIONES

Y de aquí nace la primera limitación, que por ahora tiene el *robot*: sólo analiza la *URL* introducida por el usua-

rio y no analiza recursivamente los enlaces que va encontrando en cada página (lo cual realizaremos más adelante, ya que es ésta precisamente su principal utilidad).

La segunda limitación del Robot (también en esta primera versión), es que no almacena las páginas visitadas en ninguna base de datos, ni tan siquiera en una lista encadenada. Simplemente, define la estructura necesaria para almacenar todos los datos relevantes de cada página en una lista en memoria o en registros de una base de datos. En la próxima entrega del desarrollo del *robot* utilizaremos una lista doblemente encadenada para guardar la información de las páginas analizadas.

La elección de utilizar este tipo de estructura en memoria se debe a su gran velocidad de acceso a los elementos que contiene en comparación a una base de datos. De esta forma resulta inmediato comprobar si una página ha sido ya indexada y/o analizada. Por supuesto, en cualquier momento se podrá hacer un volcado de esta lista a una base de datos. De hecho, habrá que hacer volcados periódicos, ya que un posible problema de crear una lista, es - aparte del propio mantenimiento, siempre complicado por tratarse de un manejo dinámico de datos en la memoria del ordenador -, la posibilidad de que la memoria se desborde (es decir, de que se produzca un *overflow*). La razón es que en cada página se analizan las páginas enlazadas y, de forma recursiva, se analizan a su vez los enlaces de estas páginas y así sucesivamente.

Como ya comentamos en otros artículos, este *robot* sólo indexará páginas *HTML*, por lo que no tendrá en cuenta imágenes, ficheros de audio o vídeo, direcciones de correo electrónico, etc.

Por último, y para simplificar un poco más esta primera versión, no recuperará el fichero **ROBOTS.TXT** que contiene la política del servidor *Web* sobre la posibilidad de indexación de sus páginas.

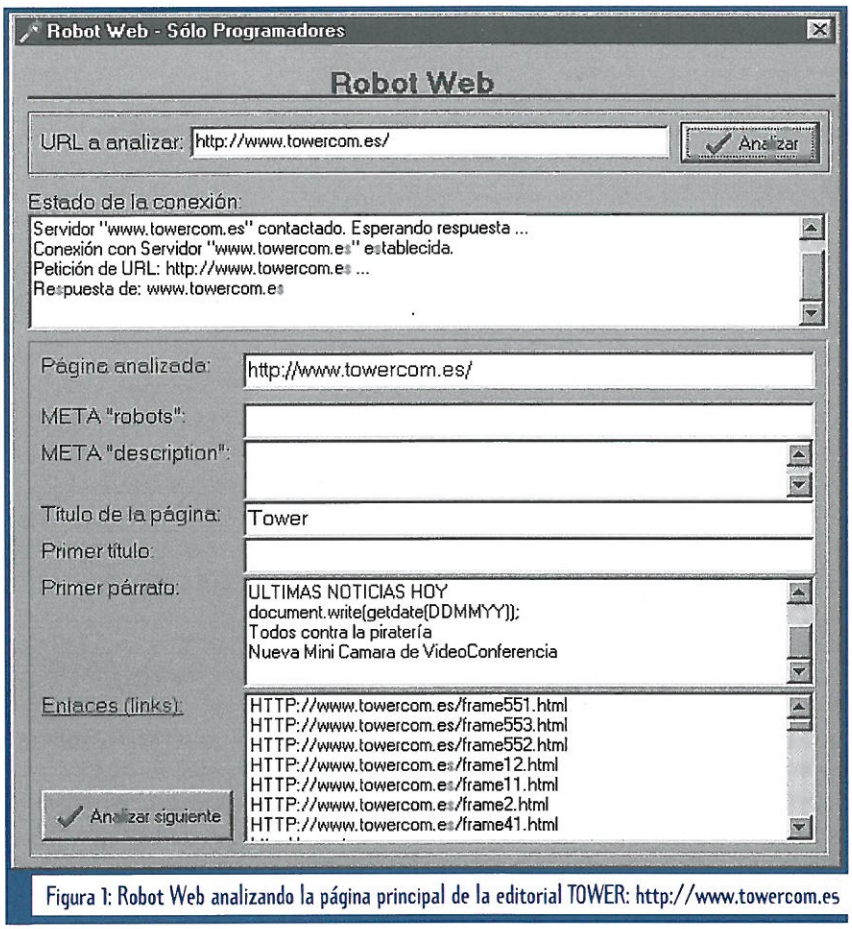


Figura 1: Robot Web analizando la página principal de la editorial TOWER: http://www.towercom.es

TIPOS Y VARIABLES

Definiremos dos estructuras de datos (de tipo *record*) que contendrán, por un lado, la información de las peticiones *HTTP* y por otro las correspondientes respuestas. Como las peticiones se realizarán sobre páginas *HTML*, juntaremos en la primera estructura los datos de la petición con los de la página *HTML* obtenida (*record*: *TPaginaHMTL*). Las cabeceras de la respuesta estarán contenidas en un *record* denominado *TRespuesta*. La razón de agrupar de esta manera la información persigue no tener una gran cantidad de variables dispersas en el código del *robot*. En el Listado 1 se muestra la estructura seguida en cada uno de estos dos registros.

Las variables declaradas en la estructura *TRespuesta* contienen sólo algunas de todas las posibles cabeceras de la respuesta. Si hace falta añadir o quitar alguna será sencillo introducir o retirar las líneas de código correspondientes.

Por su parte, en *TPaginaHTML* definimos una serie de variables que servirán para indexar cada página *HTML* y para conocer en qué estado de análisis se encuentra:

- La variable *EstadoPagina* guarda el estado, dentro de nuestro análisis particular, en que se encuentra

Listado 1. Definición de tipos utilizados, de sus variables correspondientes y de procedimientos de inicialización de las mismas.

```
  TEstadoPaginaHTML = (epNoVisitada, epIndexada, epEnAnalisis, epAnalizada);

type
  TRespuesta = record
    vProtocolo: string;
    CodigoHTTP: integer;
    CadenaHTTP: string;
    Date: string;
    Server: string;
    Allow: string;
    ContentLength: string;
    ContentType: string;
    LastModified: string;
  end;

type
  TPaginaHTML = record
    EstadoPagina: TEstadoPaginaHTML;
    Referer: string;
    Host: string;
    Address: string;
    Server: string;
    URI: string;
    Title: string;
    H1: string;
    BODY: string;

    Description: string;
    Index: boolean;
    FechaActualiz: TDateTime;
    TiempoActualiz: TDateTime;
    FrecActualiz: TDateTime;

    Follow: boolean;
    Enlaces: TStrings;
    EnlacePrimero: string;
    EnlaceAnalizado: string;
  end;

TRobotForm = class(TForm)
    RobotClientSocket: TClientSocket;
    ...
private
    { Información acerca de la página HTML, incluidos los datos de la
      propia petición }
    PaginaHTML: TPaginaHTML;

    { Información acerca de la respuesta HTTP }
    Respuesta: TRespuesta;
```

la página (como vimos en el artículo del mes pasado). El tipo de esta variable es *TEstadoPaginaHTML*.

- Las variables *Referer, Host, Address* y *URI* (*Uniform Resource Indicator,* Indicador de Recursos Uniforme), las obtenemos directamente a partir de las cabeceras de la respuesta *HTTP*.
- La variable *Server* es una variable de uso particular que almacena el nombre del *Host* o su dirección (*Address*), ya que en la petición se puede introducir el nombre de dominio o la dirección *IP*, pero no los dos a la vez. De esta manera, podemos almacenar la *URL* completa que introdujo el usuario.
- Las variables *Title, H1* y *BODY* se rellenan con la parte correspondiente de la página *HTML* (de la parte *BODY* sólo se guardarán las siete primeras líneas de texto plano, pues lo que queremos es sólo una referencia del contenido).
- Las variables *Description, Index* y *Follow* se adquieren a partir de las etiquetas *META*: *DESCRIPTION* y *ROBOTS*.
- Las variables *FechaActualiz, TiempoActualiz* y *FrecActualiz*, con significado interno sólo para el *robot* se emplean cuando la variable *Index* tiene el valor **True**, ya que indican al propio Robot cuándo se puede y debe actualizar la referencia de la página.
- Por último, las variables *Enlaces, EnlacePrimero* y *EnlaceAnalizado* sólo se utilizan cuando *Follow* el valor **True**, ya que sólo en ese caso se podrán analizar los enlaces de la página. *Enlaces* contiene la lista de los enlaces de una página, *EnlacePrimero* es el que contiene la posición, en nuestra futura lista de páginas, del primer enlace de la página y *EnlaceAnalizadoes* el que contiene la posición del último enlace anali-

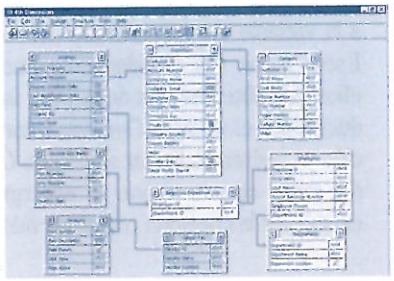
Si le gustaron otras bases de datos, 4th Dimension v6 *le sorprenderá*

4th Dimension es el entorno de bases de datos integrado más completo del mercado. Años de experiencia en el diseño de bases de datos han permitido satisfacer las siguientes necesidades:

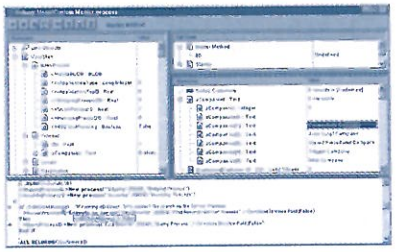
- Automatizar cualquier proceso.
- Diseñar modelos gráficos eficientes y orientados a objeto.
- Suministrar la más amplia gama de herramientas para facilitar el diseño y mantenimiento de bases de datos.
- Permitir una fácil reutilización de código.
- Permitir escalar desarrollos Cliente/Servidor, web y multiplataforma de forma transparente.

Estas son algunas de las **características** de 4th Dimension:

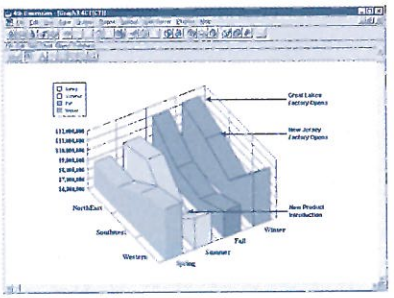
- **Motor de base de datos multiproceso, triggers, procedimientos almacenados**, presentación visual de tablas mediante diagrama de entidad /relación (ERD)



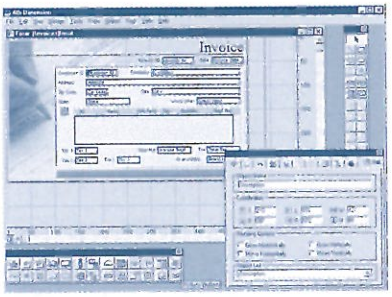
- **Debugger visual** multiproceso de última generación. **Intérprete** incluido para testeo y debugging.



- **Wizards avanzados** para diseño de formularios, queries, informes, importación, exportación, etiquetas, etc. Editor de gráficas programable 2D y 3D, soporta gráficas de barras, tartas, áreas, líneas, etc.



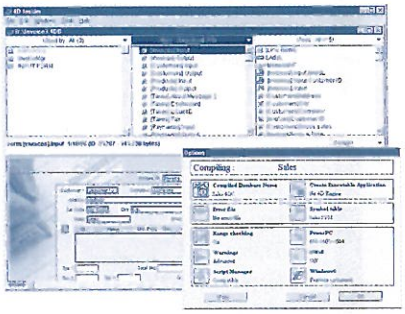
- **Editor de formularios basado en objetos.** Gestión de hojas de estilo. Redimensionamiento automático. **Docenas de objetos de interface** incluyendo check boxes, radio buttons, push buttons, áreas de scroll, picture button, botones invisibles, tab control, listas jerárquicas, termóme-tros, reglas, combo box, listas menú desplegables, menús pop-up jerárquicos, menús picture, radio pictures, listas desplegables, menús pop-up, subforms, group boxes, etc.



- Arquitectura de distribución de aplicaciones integrada y **soporte para proyectos con múltiples desarrolladores** con check-in/check-out automatizado.
- Arquitectura extensible mediante **plug-ins** y DLLs.

- **Un sólo código fuente** desplegable sin modificaciones de monousuario a 4D Server, la versión Cliente/Servidor de 4th Dimension.
- Soporte de **ODBC** (cliente y servidor) y conexiones nativas **SQL** a ORACLE, Sybase y MS SQL Server.
- **Servidor Web integrado** para acceso automático a formularios y gestión Cliente/Servidor mediante navegadores web.

- **Compilador a código máquina real** optimizado para diferentes procesadores. **Gestor gráfico de referencias cruzadas** y librerías de objetos.



VERSIÓN DEMO GRATUITA

4th Dimension es el núcleo de la gama de productos 4D. Visite http://www.ambitconsulting.com o llame al 93-209 41 11 para más información o para recibir una versión demo gratuita. Mencione este anuncio y reciba 4th Dimension v6 por el precio promocional de 34.900 Ptas*. (precio habitual: 79.000 Ptas.).

(* Actualización competitiva. Oferta válida hasta el final de existencias)

AMBIT Consulting
Balmes, 297, 4º, 2ºb. 08006 Barcelona
t. | (34) 93 209 41 11 f. | (34) 93 240 18 88
e-mail: ambit@ambitconsulting.com
http://www.ambitconsulting.com

BelSoft
Bellsoft, S.L. C/Salva, 4-2º-2ª 46002 - Valencia
t. | (34) 96 352 71 30 f. | (34) 96 352 71 30
AppleLink SPA0351 e-mail: bellsoft@combios.es
http://www.bellsoft.es

Listado 1. Continuación.

```
    procedure InicializarRespuesta;
    procedure InicializarPaginaHMTL;

...
implementation
...

procedure TRobotForm.InicializarRespuesta;
begin
  with Respuesta do begin
    vProtocolo:=   '';
    CodigoHTTP:=   0;
    CadenaHTTP:=   '';
    Date:=         '';
    Server:=       '';
    Allow:=        '';
    ContentLength:= '';
    ContentType:=  '';
    LastModified:= '';
  end;
end;

procedure TRobotForm.InicializarPaginaHMTL;
begin
  with PaginaHTML do begin
    EstadoPagina:= epNoVisitada;
    Referer:=      '';
    Host:=         '';
    Address:=      '';
    Server:=       '';
    URI:=          '';
    Description:=  '';
    Index:=        True;
    Follow:=       True;
    Title:=        '';
    H1:=           '';
    BODY:=         '';
    FechaActualiz:= 0; { '12/30/1899 12:00:00' }
    TiempoActualiz:=0;
    FrecActualiz:= 0;
    Enlaces :=     TStringList.Create;
    EnlacePrimero:= '';
    EnlaceAnalizado:= '';
  end;
end;
```

zado de la página en cada momento concreto.

Para inicializar estas estructuras de datos utilizamos dos diferentes procedimientos (ver Listado 1) , que son *InicializarRespuesta* e *InicializarPaginaHTML*.

COMENZAR EL PROCESO

Al hacer clic en el botón **Analizar** (ver Figura 1) se debe lanzar una petición *HTTP* a la *URL* escrita en el cuadro de texto: *URL a Analizar*. Lo primero, antes de nada, será establecer una conexión entre el *socket* cliente (el propio *robot*) y el *socket* servidor. Posteriormente utilizaremos esa conexión *TCP/IP* para enviar las peticiones *HTTP* y recibir las respuestas.

Antes de realizar cualquier petición HTTP tendremos que abrir una conexión TCP/IP

Para establecer la conexión entre el *socket* cliente y el servidor será necesario conocer la dirección (*Address*) o el nombre (*Host*) del servidor y asignárselo a la variable correspondiente del robot. Estos datos (en realidad sólo uno de ellos) los obtendremos de la *URL* introducida por el usuario. Con esta información ya podremos llamar al método *Open* del *robot*, que es el que se encarga de establecer la conexión con el servidor.

Una vez asignado el Host podemos llamar a la función Open de TClientSocket

El código completo del manejador del evento *OnClick* del botón *Analizar* se muestra en el Listado 2. En este listado se puede observar que aparece una llamada a un procedimiento denominado *SepararHost_y_URI*. Este procedimiento obtiene, a partir de una *URL* (*Uniform Resource Locator,* Localizador de Recursos Uniforme), el *URI* correspondiente

(es decir la localización del recurso en la estructura del servidor *Web*) y el nombre o la dirección del servidor (es decir, es capaz de distinguir si la *URL* se ha introducido con un nombre de dominio o con una dirección *IP*, y asigna de forma consecuente cada uno). El código completo del procedimiento *SepararHost_y_ URI* se puede ver en el Listado 6, en el CD-ROM.

EVENTOS GENERADOS

Al abrir la conexión con el *socket* servidor se desencadenan sucesivamente los eventos: *OnLookup, OnConnecting* y *OnConnect* (ya explicamos en el artículo del mes pasado los momentos en que se desencadena cada evento particular). En los manejadores correspondientes nos limitamos a mostrar en pantalla (en *Estado de la conexión*) cómo evoluciona la conexión con el *socket* servidor (ver Listado 3). Sólo cuando se dispare el evento *OnConnect* tendremos una conexión establecida con el servidor. Por tanto, es en ese momento cuando enviaremos la petición sobre la página seleccionada por el usuario de la aplicación. La petición de la página la implementaremos en un método aparte: *EnviarPeticionHTTP*.

PETICIÓN DE LA PÁGINA

Teniendo en cuenta el formato general de una petición *HTTP*:

```
GET /pub/ HTTP/1.1
If-Modified-Since: Thu, 15 Dic 1998 10:00:30
    GMT

Referer: http://www.towercom.es/
User-Agent: SpRobot/0.01 (Win95; I)
Host: www.towercom.es
Accept: text/html
(línea en blanco)
```

Listado 2. Procedimiento que activa el proceso de análisis del URL.

```
procedure TRobotForm.BtnAnalizarClick(Sender: TObject);
begin
 try
  InicializarPaginaHMTL;
  { Obtenemos el Host y la URI a partir de la URL y asignamos al Socket
   el Host que se va a analizar y abrimos la conexión }
  SepararHost_y_URI (EditURL.Text, PaginaHTML.Host, PaginaHTML.Address,
PaginaHTML.URI);
  RobotClientSocket.Host := PaginaHTML.Host;
  RobotClientSocket.Address := PaginaHTML.Address;
  if PaginaHTML.Host <> '' then
   PaginaHTML.Server := PaginaHTML.Host
  else
   PaginaHTML.Server := PaginaHTML.Address;
  RobotClientSocket.Open;
 except
  On E: ESocketError do begin
   MemoEstado.Lines.Delete (MemoEstado.Lines.Count - 1);
   MemoEstado.Lines.Add ('Error del Socket. Mensaje: —— ' + E.Message + ' ——
');
   MemoEstado.Lines.Add ('Servidor no encontrado. ' +
                'Revise el nombre del Host o del Servicio, o la conexión TCP/IP.');
   MemoEstado.Lines.Add
('*********************************************************');
  end;
 end;
end;
```

El código para generar la petición de una página es inmediato: sólo hay que componer un mensaje de texto con el contenido apropiado y enviarlo con el método: *SendText*. Es decir, cambiamos los valores de la petición *GET* y de la cabeceras *Host* y *Referer* por los valores de la *URL* introducida por el usuario, componemos el mensaje y lo enviamos. El código correspondiente se puede apreciar en el Listado 4.

Cuando llega la respuesta del *socket* servidor se desencadena el evento *OnRead* del *socket* cliente. Será en el manejador de este evento donde escribiremos el código que analice la respuesta.

El contenido de la respuesta lo obtenemos en el evento OnRead

Para recibir el texto utilizaremos el procedimiento *ReceiveText*, que recoge una cadena de caracteres del *Buffer* de entrada. Como la respuesta puede estar formada por una página *HTML* con muchas líneas o incluso por una respuesta *HTTP* con muchas cabeceras, se podrán disparar muchos eventos de tipo *OnRead* para una misma respuesta. Por

Listado 3. Eventos generados al abrir la conexión con el servidor asociado a la URL.

```
procedure TRobotForm.RobotClientSocketLookup(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  { Una vez abierta la conexión, se desencadenan los siguientes sucesos:
    1.- El evento OnLookup se dispara antes de intentar localizar el servidor. }
  MemoEstado.Lines.Add ('Iniciando Socket para conectar con: ' + EditURL.Text + '
...');
end;

  { 2.- El Socket se crea e inicializa para la notificación de eventos. }

procedure TRobotForm.RobotClientSocketConnecting(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  { 3.- El evento OnConnecting se dispara cuando se localiza el Socket Servidor. }
  MemoEstado.Lines.Add ('Servidor "' + PaginaHTML.Server + '" contactado. Espe-
rando respuesta ...');
end;

  { 4.- El Servidor acepta la petición de conexión y el cliente la completa. }

procedure TRobotForm.RobotClientSocketConnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  { 5.- El evento OnConnect se dispara cuando se establece la conexión. }
  MemoEstado.Lines.Add ('Conexión con Servidor "' + Socket.RemoteHost + '" esta-
blecida.');
  MemoEstado.Lines.Add ('Petición de URL: http://' + EditURL.Text + ' ...');

  { Enviamos la petición de la URL }
  EnviarPeticionHTTP (Socket);
end;
```

Listado 4. Procedimiento para enviar la petición al servidor Web.

```
procedure TRobotForm.EnviarPeticionHTTP (Socket: TCustomWinSocket);
var
  Mensaje: string;
begin
  Mensaje := 'GET ' + PaginaHTML.URI + ' HTTP/1.0' + #13#10;
  Mensaje := Mensaje + 'Referer: ' + PaginaHTML.Referer + #13#10;
  Mensaje := Mensaje + 'Connection: Close' + #13#10;
  Mensaje := Mensaje + 'User Agent: SpRobot/0.01 (Win95; I)' + #13#10;
  Mensaje := Mensaje + 'Host: ' + PaginaHTML.Host + #13#10;
  Mensaje := Mensaje + 'Accept: text/html' + #13#10;
  Mensaje := Mensaje + #13#10;

  { Enviamos todo }
  Socket.SendText (Mensaje);
end;
```

tanto, necesitaremos una variable auxiliar que distinga si la respuesta proviene de la misma conexión que la anterior o de una nueva. Esta variable que denominaremos: *SockHandle*, almacenará el descriptor del *socket* que atiende la conexión y servirá para comprobar si el contenido del *Buffer* de entrada corresponde a la misma transferencia o a una nueva.

Cuando se dispare el evento OnConnect estará establecida la conexión con el servidor

En el código del manejador del evento *OnRead* se pueden observar las llamadas a los dos procedimientos que se encargan de analizar la respuesta. El primero de ellos, denominado *ProcesarCabecerasRespuesta* se encarga de analizar las cabeceras de la respuesta *HTTP*. Su formato es muy parecido al que utilizamos en su momento - para analizar las cabeceras de la petición - cuando desarrollamos el *Servidor Web* (ver revistas de *Sólo Programadores* números 45 a 49).

El segundo procedimiento *ProcesarPaginaHTML* realiza el análisis e indexación de la página *HTML* devuelta por el servidor *Web*. El código de ambos procedimientos se muestra en el Listado 5 en el CD-ROM.

PROCESAR LA PÁGINA HTML

Para analizar la página *HTML* sólo tenemos que buscar las etiquetas con la información que es relevante para nuestro *robot*. En concreto, debemos buscar las etiquetas *META ROBOTS* y *DESCRIPTION*, la etiqueta del título de la página (*<TITLE>*), la etiqueta del primer título dentro de la

página (*<H1>* ó *<H2>*), la etiqueta del cuerpo de la página (*<BODY>*) y las etiquetas que contienen los enlaces a otras páginas (*<A HREF="...">*).

Para determinar el final de la respuesta HTTP utilizamos el descriptor del socket

Cuando la página haya sido indexada deberemos cambiar el "estado" de la página al valor **epIndexada**. Para encontrar el contenido de las etiquetas implementaremos unas funciones específicas para este propósito. Como las etiquetas *META* tienen un formato distinto al resto, y como ocurre otro tanto con las etiquetas que contienen enlaces a otras páginas, crearemos tres procedimientos: uno para obtener etiquetas *META*, otro para obtener los enlaces a otras páginas y otro para obtener el resto de etiquetas. Estos métodos son respectivamente:

BuscarEtiquetaMETA
BuscarEnlaces
BuscarEtiqueta

y el listado de su código aparece en el Listado 6 del CD-ROM.

No vamos a entrar a describir en detalle estos últimos procedimientos ya que lo único que hacen es filtrar el código en busca de patrones y obtener el resultado deseado. Tan sólo es necesario realizar unas reseñas:

- Del cuerpo de la página sólo se obtienen las siete primeras líneas de texto plano.
- Si no se encuentra ninguna etiqueta *<H1>* se busca una etiqueta *<H2>*
- A los enlaces relativos (es decir, que sólo contienen la *URI*) se les añade el nombre o la dirección del servidor y el protocolo (**http://**).

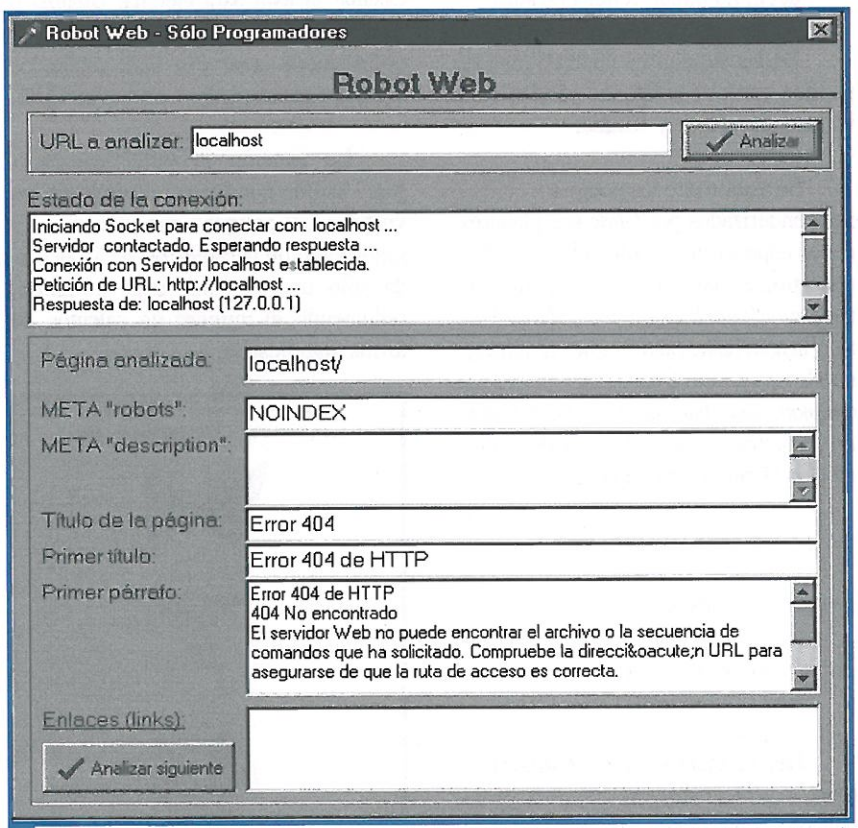

Robot Web - Sólo Programadores
Robot Web
URL a analizar: localhost
Analizar
Estado de la conexión:
Iniciando Socket para conectar con: localhost ...
Servidor contactado. Esperando respuesta ...
Conexión con Servidor localhost establecida.
Petición de URL: http://localhost ...
Respuesta de: localhost (127.0.0.1)
Página analizada: localhost/
META "robots": NOINDEX
META "description":
Título de la página: Error 404
Primer título: Error 404 de HTTP
Primer párrafo: Error 404 de HTTP
404 No encontrado
El servidor Web no puede encontrar el archivo o la secuencia de comandos que ha solicitado. Compruebe la dirección URL para asegurarse de que la ruta de acceso es correcta.
Enlaces (links):
Analizar siguiente

Figura 2. Resultado de error obtenido por el robot Web al intentar recuperar la página principal del Host Local (http://localhost) sin haber determinado ésta en el servidor.

En la Figura 1 se muestra el resultado que obtiene nuestro *robot Web* al lanzar una petición a la página principal de la editorial *Tower* **http://www.towercom.es.**

Para realizar las pruebas, sin embargo, no necesitamos tener acceso a Internet. Si instalamos un servidor *Web* en nuestro ordenador (por ejemplo *Personal Web Server* de *Microsoft* o incluso el servidor *Web* que desarrollamos en esta revista), podemos lanzar una petición al *host l*ocal (**http://localhost**, ó **http://127.0.0.1**) y comprobar el funcionamiento de nuestro *robot*. En concreto, se puede ver en la Figura 2 el resultado de la respuesta *HTTP* a una petición enviada a la página principal del *host* local, sin haber determinado previamente esta página en el servidor *Web* (que para más señas, era en este caso el *Personal Web Server*). Se puede observar que el código devuelto por el servidor es el **404**, es decir: página no encontrada en el servidor. El ejecutable del robot se puede encontrar en el CD-ROM de la revista (**SpRobot.exe**), junto con los listados indicados a lo largo del artículo.

CONCLUSIÓN

Hemos generado un *robot Web* que cumple la función básica de analizar todo el contenido de una página *HTML* que le pasamos como parámetro. En los sucesivos artículos ampliaremos la función de este *robot*, para que mantenga una lista de las páginas por las que vaya pasando y almacene en una Base de Datos toda la información relevante de las mismas. Posteriormente integraremos todo este desarrollo en el proyecto de nuestro Buscador *Web*.

Glide (III)

Constantino Sánchez Ballesteros (constantino@nexo.es)

Continuamos nuestros balbuceos en *Glide* aprendiendo las normas generales para construir polígonos, representar técnicas de antialiasing (suavizado de bordes de un polígono), especificación de colores, etc.

POLÍGONOS CONVEXOS

Un polígono es un área plana encerrada por segmentos de líneas que se unen a su vez por vértices. Dado que las tarjetas *Voodoo* no renderizan polígonos directamente, *Glide* provee una serie de funciones de renderizado que se han optimizado para su uso directo con el hardware. Los polígonos renderizados por las funciones de *Glide* están sujetos a algunas restricciones determinadas:

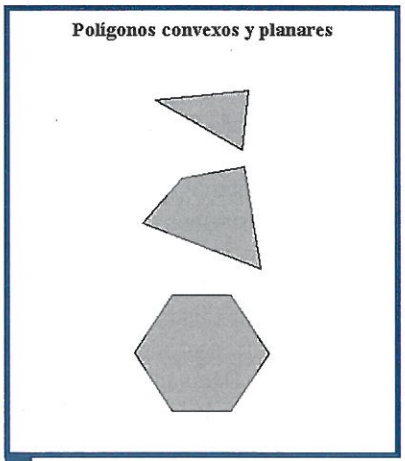


Figura 1. Polígonos válidos en Glide.

- Las esquinas del polígono no pueden interseccionarse entre sí.
- El polígono debe ser convexo, es decir, no puede tener entradas.

En las siguientes ilustraciones se pueden apreciar polígonos válidos y no válidos para *Glide* y *Voodoo*:

Se asume que los polígonos convexos renderizados por *Glide* son planares en el espacio de coordenadas. Existen dos rutinas para renderizar polígonos (*grDrawPlanarPolygon()* y *grDrawPlanarPolygonVertexList()*) que requieren que los parámetros de render (*r*, *g*, *b*, *a*, *ooz*, *oow*, *sow*, *tow*) también sean planares. Ninguna de las funciones de *render* efectúa algún tipo de *clipping*.

```
void grDrawPlanarPolygon(int nVerts, int ilist[],
      const GrVertex vlist[])
void grDrawPlanarPolygon( int nVerts, int ilist[],
      const GrVertex vlist[] )
void grDrawPolygon( int nVerts, int ilist[],
      const GrVertex vlist[] )
```

Las funciones *grDrawPlanarPolygon()* y *grDrawPolygon()* renderizan un polígono convexo con *nVerts* vértices. El segundo argumento, *ilist* es un array de índices dentro de la lista de vértices provistos en el tercer argumento. Este nivel de indirección en la especificación de vértices es útil cuando se necesita precalcular la lista para efectuar *clipping* o eliminar caras no visibles. El preprocesador puede crear *ilist* para copiar los vértices seleccionados a otra nueva lista.

La función *grDrawPlanarPolygon()* asume que los parámetros de los vértices del polígono son planares. La graduación de parámetros será calculada sólo una vez por cada polígono, reduciendo el número de cálculos de forma significativa.

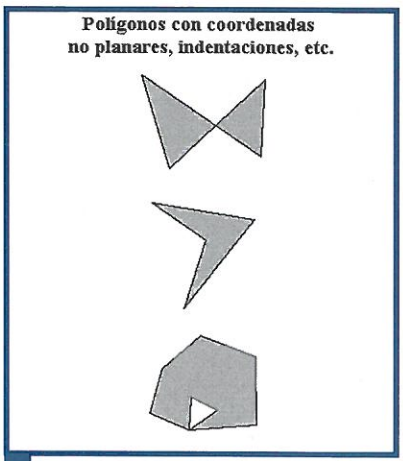


Figura 2. Polígonos no válidos en Glide.

Otro par de funciones de *render* de polígonos definidas en *Glide*, *grDrawPlanarPolygonVertexList()* y *grDrawPolygonVertexList()* son funcionalmente equivalentes a *grDrawPlanarPolygon ()* y *grDrawPolygon()*, respectivamente. La única diferencia entre estas funciones es la forma en que se especifican los vértices.

```
void grDrawPlanarPolygonVertexList
     ( int nVerts, const GrVertex vlist[] )
void grDrawPolygonVertexList( int nVerts,
     const GrVertex vlist[] )
```

No existe nivel de indirección en *grDrawPlanarPolygonVertexList()* y *grDrawPolygonVertexList()*. El vértice i^{th} del polígono pasado a estas rutinas es *vlist[I]*, asumiendo que $0 \le I < nVerts$, donde el vértice i^{th} de un polígono pasado a *grDrawPlanarPolygon ()* ó *grDrawPolygon()* es *vlist[ilist[I]]*.

BACKFACE CULLING (SELECCIÓN DE CARAS OCULTAS)

Glide soporta *backface culling* basado en el signo de un área de un polígono. Cuando *Glide* renderiza un polígono, el primer paso consiste en dividir dicho polígono en triángulos. En la siguiente ilustración se muestran un par de triángulos cuyos vértices han sido etiquetados de forma acorde a la regla dada anteriormente:

Los polígonos de la izquierda de la figura se han definido en relación a un origen en la esquina superior izquierda, mientras que los de la derecha tienen el origen en la esquina inferior izquierda. *Clockwise* y *Counter-Clockwise* (rotación en el sentido de las agujas del reloj) se refieren a la dirección en que los vértices son movidos en orden alfabético.

El signo del área del triángulo puede ser utilizado para efectuar el *backface culling* (eliminación de triángulos que no serán visibles en pantalla antes de renderizarlos). Dado que el área debe ser calculada, se trata de una forma poco costosa de eliminar caras. De todos modos, eliminar los triángulos más cercanos puede ser más beneficioso. Por ejemplo, si el *backculling* se efectúa antes que la representación de luces, los cálculos para visualizar la luz sobre los triángulos se pueden eliminar.

Los polígonos en Glide deben ser convexos

La función *Glide grCullMode()* sólo tiene un parámetro, un modo que puede ser asignado como *GR_CULL_NONE*, *GR_CULL_ NEGATIVE* ó *GR_CULL_ POSITIVE*. Cuando el modo de *culling* es el primero (valor por defecto), todos los polígonos son renderizados en la pantalla menos los correspondientes al signo para ese área. Si el signo del área se corresponde con *mode*, el triángulo será ignorado. *grCullMode()* asume que *GR_CULL_POSITIVE* se corresponde a una orientación *counter-clockwise* cuando el origen aparece en la esquina superior izquierda ; se corresponderá a una orientación *clockwise* cuando el origen se sitúe en la esquina inferior izquierda.

```
void grCullMode( GrCullMode_t mode )
```

grCullMode() no tiene efecto sobre puntos o líneas, pero sin embargo,si que afecta al renderizado de los triángulos y polígonos.

ANTIALIASING

Si observamos detenidamente las líneas que se representan en la pantalla notaremos que suelen ofrecer una especie de dientes en su visualización, sobre todo las líneas no horizontales y no verticales. Este hecho es conocido como *aliasing*. Para resolver este problema se utiliza la técnica de *antialising* que consiste en dibujar *pixels* adyacentes con un tono de color diferente para lograr suavizar el aspecto de la línea en la pantalla. En la siguiente ilustración podemos apreciar *aliasing* y *antialiasing*.

Para observar con más detalle el proceso de *antialiasing*, en la figura 5 se puede apreciar la construcción de una línea (parte derecha) suavizada mediante esta técnica:

Glide dibuja puntos suavizados, líneas, triángulos y polígonos definiendo el factor *alpha* de los bordes de estos objetos. Por ello, debemos configurar correctamente la unidad de combinación *alpha* y activar *alpha blending* antes de utilizar cualquier comando de dibujo que utilice *antialiasing*:

- Establecer la unidad de combinación *alpha* para producir el efecto buscado.
- Asignar la función de *alpha blending*. Estas funciones se especifi-

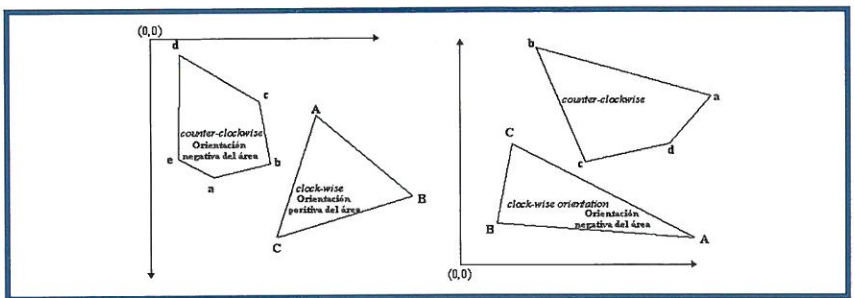


Figura 3. Orientación de un polígono y señalización del área.

Tabla 1. Localización de la orientación del triángulo.

La localización del origen afecta a la orientación del triángulo y al signo de su área.

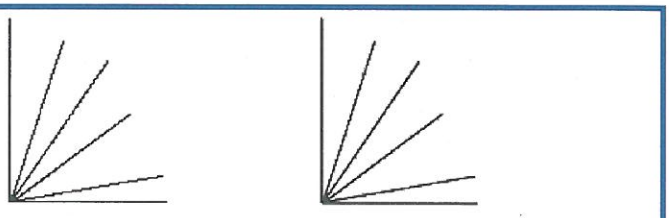
Si la localización del origen es...	y la orientación del triángulo es...	entonces el signo del área será
GR_ORIGIN_LOWERLEFT	Clockwise	Negativo
GR_ORIGIN_LOWERLEFT	Counter-clockwise	Positivo
GR_ORIGIN_UPPERLEFT	Clockwise	Positivo
GR_ORIGIN_UPPERLEFT	Counter-clockwise	Negativo

can para los componentes del color fuente y destino y para los valores *alpha* del color fuente y destino. La elección depende de si la escena se renderiza *front to back* ó *back to front* (de adelante/atras o de atrás/adelante, respectivamente).

- Establecemos el valor *alpha* para cada vértice. El valor *alpha* elegido representa la transparencia del objeto que está siendo renderizado; con objetos opacos asignaremos el valor **255**. Este valor *alpha* se multiplicará por el *pixel* para obtener el valor final *alpha* utilizado para el *alpha blending*.
- Llamaremos a *grAADraw* ó *grAADraw*. Estas funciones se detallan a continuación:

```
void grAADrawPoint( GrVertex *p )

void grAADrawLine( GrVertex *va,
      GrVertex *vb )
void grAADrawTriangle(      GrVertex *va,
      GrVertex *vb, GrVertex *vc, FxBool
      aaAB, FxBool aaBC, FxBool aaCA )

void grAADrawPolygon( int nVerts, const int
      ilist[], const GrVertex vlist[] )

void grAADrawPolygonVertexList( int nVerts,
      const GrVertex vlist[] )

void guAADrawTriangleWithClip( const GrVer-
      tex *va, const GrVertex *vb, const
      GrVertex *vc )
```

- *grAADrawPoint()* renderiza un punto como cuatro *pixels*, cada uno suavizado acorde con el *pixel* principal. Las líneas dibujadas con *grAADrawLine()* aparecerán algo más gruesas de lo esperado.
- *grAADrawTriangle()* tiene tres argumentos más que *grDrawTriangle()*. Estos argumentos, *aaAB*, *aaBC*, y *aaBC* son valores booleanos que permiten a las esquinas del triángulo ser suavizadas de forma selectiva.
- *grAADrawPolygon()* y *grAADrawPolygonVertexList()* dibuja polígonos convexos con esquinas suavizadas.
- *guAADrawTriangleWithClip()* crea un *clipping* 2D del triángulo especificado y dibuja el polígono resultante con *grAADrawPolygonVertexList()*. Todas las esquinas del triángulo serán suavizadas.



(a) Las líneas con Aliasing simulan tener "dientes de sierra"

(b) Con Antialiasing se suavizan las esquinas mediante fundido de píxels más cercanos

Figura 4. Detalle de aliasing y antialiasing.

Los objetos que se dibujen en la pantalla deben ser colocados por orden de profundidad (*depth*). Los factores de *alpha blending* dependen de si la escena es colocada *front to back* ó *back to front*. El primer ejemplo muestra los factores de *alpha blending* si la escena se dibuja como *front to back*.

```
/* Establecemos unidad de combinación alpha
      para producir el valor alpha */
grAlphaCombine(GR_COMBINE_SCALE_OTH
      ER, GR_COMBINE_FACTOR_ONE,
      GR_LOCAL_NONE,
      GR_LOCAL_INTERATED, FXFALSE);

/* Suavizamos colores basados en el valor alpha */
grAlphaBlendFunction(GR_BLEND_
      ALPHA_SATURATE, GR_BLEND_ONE,
      GR_BLEND_ SATURATE,
      GR_BLEND_ONE);

/* Dibujamos la escena utilizando las rutinas
      grAADraw */
```

Debemos sustituir los factores de *alpha blending* mostrados a continuación si la escena se va a dibujar como *back to front*:

```
GrAlphaBlendFunction(GR_BLEND_SRC_
      ALPHA,
      GR_BLEND_ONE_MINUS_SRC_ALPHA,
       GR_BLEND_ZERO,
      GR_BLEND_ZERO);
```

ESPECIFICANDO COLORES

Un color consiste en tres o cuatro componentes diferentes de color que son:

1. Rojo.
2. Verde.
3. Azul.
4. *Alpha*.

Los valores del componente de color deben estar en el rango [0..255] donde **0** equivale al color negro y **255** la máxima intensidad (blanco). Los componentes de color son agrupados para crear un color determinado. *Glide* soporta cuatro clasificaciones diferentes para los bytes de color, definidos en el tipo enumerado *GrColorFormat_t*. La clasificación del *byte* de color determina cómo trabaja el *frame buffer* y los argumentos de color son interpretados y establecidos en la llamada a la función **grSstWinOpen()** cuando el sistema *Glide* y *Voodoo* son inicializados. La definición de tipo *GrColor_t* representa un valor de color agrupado y es utilizado en las rutinas que asignan un color constante:

- *grBufferClear()*
- *grConstantColorValue()*
- *grFogColorValue()*
- *grChromakeyValue()*

```
void grConstantColorValue( GrColor_t color )
```

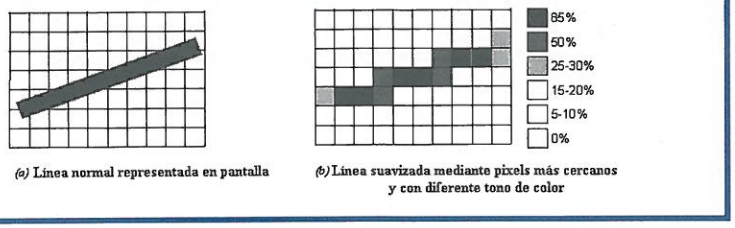


Figura 5. Técnica de antialiasing.

Glide se refiere a un color constante cuando se crea renderizado con sombreado flat, mediante la función *grConstantColorValue()*. El valor es **0x FFFFFFFF**. Los colores de los vértices se especifican en la estructura *GrVertex*.

DITHERING

El hardware de las tarjetas *3DFX* representa internamente el color como valores de 32 *bits* en un formato especificado en la llamada a la función *grSstWinOpen()*. Este color es eventualmente cambiado a 16 *bits* para guardarlo en el *frame buffer*, y posteriormente es filtrado a 24 *bits* para efectuar la representación final. Desde la perspectiva de una aplicación, este cambio de formato resulta transparente. El *Dithering* es una técnica que permite incrementar el rango de colores percibidos en una imagen aplicando un patrón de *pixels* cercanos que modifican sus valores de color. El *Dithering* es similar al medio tono utilizado en las publicaciones en blanco y negro que producen sombras de grises.

```
void grDitherMode( GrDitherMode_t mode )
```

La función *grDitherMode()* selecciona la forma de *dithering* que utilizará la *Voodoo* cuando convierta valores de 24 *bits RGB* a 16 *bits RGB*. Los valores válidos para esta función son *GR_DITHER_DISABLE*, *GR_DITHER_2x2*, y *GR_DITHER_4x4*, siendo este último el modo de *Dither* por defecto.

Tabla 2. Combinaciones de color.

Función de combinación de color	Color computado
GR_COMBINE_FUNCTION_ZERO	
GR_COMBINE_FUNCTION_LOCAL	clocal
GR_COMBINE_FUNCTION_LOCAL_ALPHA	alocal
GR_COMBINE_FUNCTION_SCALE_OTHER GR_COMBINE_FUNCTION_BLEND_OTHER	f * cother
GR_COMBINE_FUNCTION_SCALE_OTHER _ADD_LOCAL	f * cother + clocal
GR_COMBINE_FUNCTION_SCALE_OTHER _ADD_LOCAL_ALPHA	f * cother + alocal
GR_COMBINE_FUNCTION_SCALE_OTHER _MINUS_LOCAL	f * (cother – clocal)
GR_COMBINE_FUNCTION_SCALE_OTHER _MINUS_LOCAL_ADD_LOCAL GR_COMBINE_FUNCTION_BLEND	f * (cother – clocal) + clocal∫ f * cother + (1 – f) * clocal
GR_COMBINE_FUNCTION_SCALE_OTHER _MINUS_LOCAL_ADD_LOCAL_ALPHA	f * (cother – clocal) + alocal
GR_COMBINE_FUNCTION_SCALE_MINUS _LOCAL_ADD_LOCAL GR_COMBINE_FUNCTION_BLEND_LOCAL	f * (– clocal) + clocal ∫ (1 – f) * clocal
GR_COMBINE_FUNCTION_SCALE_MINUS _LOCAL_ADD_LOCAL_ALPHA	f * (– clocal) + alocal

LA UNIDAD DE COMBINACIÓN DE COLOR

El control de las funciones de *rendering* de alto nivel es controlado por tres funciones:

1. *grColorCombine()*
2. *grAlphaCombine()*

Tabla 3. Factores de escala.

Factor de combinación	Factor de escala (f)
GR_COMBINE_FACTOR_NONE	No especificado
GR_COMBINE_FACTOR_ZERO	0
GR_COMBINE_FACTOR_LOCAL	clocal / 255
GR_COMBINE_FACTOR_OTHER_ALPHA	aother / 255
GR_COMBINE_FACTOR_LOCAL_ALPHA	alocal / 255
GR_COMBINE_FACTOR_TEXTURE_ALPHA	atexture / 255
GR_COMBINE_FACTOR_ONE	1
GR_COMBINE_FACTOR_ONE_MINUS_LOCAL	1 – clocal / 255
GR_COMBINE_FACTOR_ONE_MINUS_OTHER_ALPHA	1 – aother / 255
GR_COMBINE_FACTOR_ONE_MINUS_LOCAL_ALPHA	1 – alocal / 255
GR_COMBINE_FACTOR_ONE_MINUS_TEXTURE_ALPHA	1 – atexture / 255

3. grTexCombine()

Mientras estas tres funciones sean representadas individualmente, los valores de una función pueden afectar de forma significativa la entrada de otras rutinas. La unidad de combinación de color calcula un color *RGB* por cada *pixel* que sea renderizado. Los valores introducidos se pueden sumar, suavizar y/o escalar para producir sombreado *Flat/Gouraud* con efectos opcionales de luz. Esta unidad de color calcula cada color de forma independiente, pero los tres colores son calculados utilizando la misma fórmula. Esta unidad también gestiona el valor *alpha*. La unidad de combinación de color gestiona un componente de color mediante la fórmula:

```
c = f * a + b
```

donde *c* es el componente rojo, verde o azul, *f* es el factor de escala y *a b* son sumandos para diferenciar las elecciones de entrada. La función Glide que configura la unidad de combinación de color es *grColorCombine()* y se encarga de especificar la función que computa el color, así como de seleccionar la entrada.

```
void grColorCombine(GrCombineFunction_t func,
      GrCombineFactor_t     factor,
      GrCombineLocal_t      local,
      GrCombineOther_t      other,
      FxBool   invert
)
```

En la siguiente tabla se muestran los catorce valores posibles para las combinaciones de color. El primer argumento de *grColorCombine()*, *func*, especifica el color. La columna de la derecha representa la función de combinación que corresponde a cada nombre simbólico. *F* es el factor de escala y es definido en el argumento *factor* de *grColorCombine()*. C_{local} y c_{other} son especificados por el tercer y cuarto argumentos. Algunas de las fórmulas especifican un valor *alpha* ($?_{local}$) que es definido en la función *grAlphaCombine()*. El segundo argumento de *grColorCombine()*, *factor*, especifica el factor de escala tal y como se ha comentado anteriormente. Su valor es elegido por la cantidad de símbolos listados en la columna de la izquierda de la siguiente tabla. La columna de la derecha da el factor de escala que corresponde a cada nombre simbólico. $\mathbf{C}_{local}$ se especifica mediante el tercer argumento de *grColorCombine()*, $\mathbf{a}_{local}$ y $\mathbf{a}_{other}$ son definidos en la función *grAlphaCombine()*, y $\mathbf{a}_{texture}$ procede de la unidad de combinación de texturas.

El tercer y cuarto argumentos de *grColorCombine()* establecen valores para las variables c_{local} y c_{other} que aparecen en las funciones de combinación; las elecciones se muestran en la tabla de abajo. Los colores se calculan mediante la iteración de los especificados en las estructuras *GrVertex* pasadas en las operaciones de dibujo. El color de la textura procede de la unidad de combinación de texturas y el color constante es establecido por *grConstantColorValue()*.

La fórmula *func* calcula los componentes de color rojo, verde y azul. El resultado de este cálculo entra dentro del rango [0..255] y puede ser invertido, basado en el argumento final de *grColorCombine()* (*invert*). Invertir los *bits* en un componente de color *c* es lo mismo que establecer **(1.0 – c)** para valores en coma flotante dentro del rango [0..1] ó (255 – *c*) para valores de 8 *bits* dentro del rango [0..255].

La unidad de combinación de color computa el color fuente (*Source*), siendo la combinación por defecto del color la siguiente:

```
grColorCombine(GR_COMBINE_FUNC-
      TION_SCALE_OTHER,
      GR_COMBINE_FACTOR_ONE,
```

Tabla 4. Elección de colores.

Combinación local fuente (source)	color local (clocal)
GR_COMBINE_LOCAL_NONE	Color no especificado
GR_COMBINE_LOCAL_ITERATED	Color del vértice (sombreado Gouraud)
GR_COMBINE_LOCAL_CONSTANT	Color constante
Otra combinación fuente	**Otro color (cother)**
GR_COMBINE_OTHER_NONE	Color no especificado
GR_COMBINE_OTHER_ITERATED	Color del vértice (sombreado Gouraud)
GR_COMBINE_OTHER_TEXTURE	Color del mapa de texturas
GR_COMBINE_OTHER_CONSTANT	Color constante

```
    GR_COMBINE_LOCAL_ITERATED,
    GR_COMBINE_OTHER_ITERATED
    FXFALSE )
```

DIBUJAR CON UN COLOR CONSTANTE

A continuación se mostrarán diversos extractos de código mostrando combinaciones de color con sombreados. Este primer código dibuja un triángulo coloreado asignando un color constante e indicando a la unidad de combinación de color que lo utilice como c_{other}.

```
GrVertex a, b, c;

/* Asignamos un color constante (se asume el
      formato ARGB) */
grConstantColorValue( (100<<8) + 150 );

/* configuramos la unidad de color para el color
      constante */
grColorCombine(GR_COMBINE_FUNC-
      TION_SCALE_OTHER, GR_COMBI-
      NE_FACTOR_ONE,
      GR_COMBINE_LOCAL_NONE,
      GR_COMBINE_OTHER_CONSTANT,
      FXFALSE);
/* se asume que algunas coordenadas se han
      asignado a las variables a b c */
grDrawTriangle(&a, &b, &c);
```

En el siguiente código se obtiene el mismo resultado que en el listado anterior, pero utilizando c_{local} como color constante:

```
GrVertex a, b, c;
/* establecemos color */
grConstantColorValue( (100<<8) + 150);

/* Configuramos unidad de combinación*/
grColorCombine(GR_COMBINE_FUNC-
      TION_LOCAL, GR_COMBINE_FAC-
      TOR_NONE,
      GR_COMBINE_LOCAL_CONSTANT,
      GR_COMBINE_OTHER_NONE,
      FXFALSE);

/* dibujamos el triángulo */
grDrawTriangle(&a, &b, &c);
```

SOMBREADO FLAT

En el siguiente ejemplo se dibuja un triángulo con sombreado *Flat* utilizando el color para el vértice A. Se asigna el color constante al color del vértice y se procede como en el ejemplo anterior.

```
GrVertex A, B, C;

/* Asginamos un color constante al color del
      vértice A (se asume ARGB) */
grConstantColorValue((((int)A.a)<<24)|(((int)
      A.r)<<16)|(((int)A.g)<<8)|(int) A.b);

/* Configuramos unidad de combinación de color
      para el color constante */
grColorCombine(GR_COMBINE_FUNC-
      TION_LOCAL, GR_COMBINE_FAC-
      TOR_NONE,
      GR_COMBINE_LOCAL_CONSTANT,
      GR_COMBINE_OTHER_NONE,
      FXFALSE);
/* Dibujamos el triángulo */
grDrawTriangle(&A, &B, &C);
```

De forma alternativa podríamos asignar los colores de los tres vértices a los colores en el vértice A y proceder como en el siguiente ejemplo:

```
GrVertex A, B, C;

/* establecemos todos los vértices al mismo
      color */
B.a = C.a = A.a;
B.r = C.r = A.r;
B.g = C.g = A.g;
B.b = C.b = A.b;

/* Configuramos la unidad de combinación para
      los colores iterados */
grColorCombine(GR_COMBINE_FUNC-
      TION_LOCAL, GR_COMBINE_FAC-
      TOR_NONE,
      GR_COMBINE_LOCAL_ITERATED,
      GR_COMBINE_OTHER_NONE,
      FXFALSE);

/* dibujamos triángulo */
grDrawTriangle(&A, &B, &C);
```

SOMBREADO GOURAUD

A continuación procederemos con el sombreado *Gouraud*, el cual ofrece más calidad visual en la representación de las escenas. Este sombreado se realiza suavizando el color de vértice a vértice. El hardware iterará automáticamente los colores para obtener dicho sombreado. La unidad de combinación de color está configurada con c_{local} asignada a los componentes de color iterados:

```
GrVertex a, b, c;

/* configure color combine unit for iterated color
      */
grColorCombine(GR_COMBINE_FUNC-
      TION_LOCAL, GR_COMBINE_FAC-
      TOR_NONE,
      GR_COMBINE_LOCAL_ITERATED,
      GR_COMBINE_OTHER_NONE,
      FXFALSE);

/* assumes that some coordinates have been
      assigned to a, b, and c */
grDrawTriangle(&a, &b, &c);
```

De forma alternativa c_{other} puede ser dirigida a los componentes de color iterados:

```
GrVertex a, b, c;

/* configuramos unidad para colores iterados */
grColorCombine(GR_COMBINE_FUNC-
      TION_SCALE_OTHER, GR_COMBI-
      NE_FACTOR_ONE,
      GR_COMBINE_LOCAL_NONE,
      GR_COMBINE_OTHER_ITERATED,
```

```
        FXFALSE);

/* dibujamos triángulo */
grDrawTriangle(&a, &b, &c);
```

SOMBREADO FLAT CON TEXTURAS

El siguiente ejemplo produce un triángulo texturado en modo *Flat* utilizando un color constante:

```
GrVertex a, b, c;

/* asignamos color */
grConstantColorValue( (100<<8) + 150);

/* configuramos unidad de combinación */
grColorCombine(GR_COMBINE_FUNC-
        TION_SCALE_OTHER, GR_COMBI-
        NE_FACTOR_LOCAL, GR_COMBI-
        NE_LOCAL_CONSTANT, GR_COMBI-
        NE_OTHER_TEXTURE, FXFALSE);

/* Dibujamos triángulo */
grDrawTriangle(&a, &b, &c);
```

TRIÁNGULO TEXTURADO CON SOMBREADO GOURAUD

Este ejemplo configura la unidad para el sombreado *Gouraud* con texturas dirigiendo c_{local} al color iterado y c_{other} a la salida de la unidad de combinación de texturas.

```
GrVertex a, b, c;

/* Configuramos unidad de color para el color
        iterado */
grColorCombine(GR_COMBINE_FUNC-
        TION_SCALE_OTHER, GR_COMBI-
        NE_FACTOR_LOCAL, GR_COMBI-
        NE_LOCAL_ITERATED, GR_COMBI-
        NE_OTHER_TEXTURE, FXFALSE);

/* dibujamos triángulo */
grDrawTriangle(&a, &b, &c);
```

SOMBREADO GOURAUD CON LUZ ESPECULAR

Este ejemplo produce un triángulo texturado con luz especular provista en la iteración del color *RGB*:

```
GrVertex a, b, c;

/* Configura unidad de combinación de color */
grColorCombine(GR_COMBINE_FUNC-
        TION_SCALE_OTHER_ADD_LOCAL,
        GR_COMBINE_FACTOR_ONE,
        GR_COMBINE_LOCAL_ITERATED,
        GR_COMBINE_OTHER_TEXTURE,
        FXFALSE);

/* Dibuja el triángulo */
grDrawTriangle(&a, &b, &c);
```

TRIÁNGULO TEXTURADO CON LUZ Y BRILLO ESPECULAR

Utilizando el componente *alpha* a un modelo monocromo con luz especular y brillo, podemos producir triángulos que reflejan brillo (textura *RGB* * *RGB*) + *alpha*):

```
GrVertex a, b, c;

/* Configuramos unidad para el color iterado */
grColorCombine(GR_COMBINE_FUNC-
        TION_SCALE_OTHER_ADD_LOCAL_A
        LPHA,
        GR_COMBINE_FACTOR_LOCAL,
        GR_COMBINE_LOCAL_ITERATED,
        GR_COMBINE_OTHER_TEXTURE,
        FXFALSE);

/* dibujamos el triángulo */
grDrawTriangle(&a, &b, &c);
```

MONOCROMO DIFUSO

Creamos un triángulo con las características anteriores más la obtención del difuso en el modelo monocromo (textura *RGB* * *alpha* + *RGB*. *Alpha* es elegido mediante a_{local} ó a_{other} con una llamada a *grAlphaCombine()* que no se muestra en el ejemplo. En el siguiente segmento de código *alpha* se asume como a_{local}:

```
GrVertex a, b, c;

grColorCombine(GR_COMBINE_FUNC-
        TION_SCALE_OTHER_ADD_LOCAL,
        GR_COMBINE_FACTOR_LOCAL_ALP
        HA, GR_COMBINE_LOCAL_ITERA-
        TED, GR_COMBINE_OTHER_TEXTU-
        RE, FXFALSE);

grDrawTriangle(&a, &b, &c);
```

De forma alternativa, *alpha* puede especificarse como a_{other} en *grAlphaCombine()*. En ese caso se necesita la configuración *grColor Combine()*.

```
GrVertex a, b, c;

grColorCombine(GR_COMBINE_FUNC-
        TION_SCALE_OTHER_ADD_LOCAL,
        GR_COMBINE_FACTOR_OTHER_ALP
        HA, GR_COMBINE_LOCAL_ITERA-
        TED, GR_COMBINE_OTHER_TEXTU-
        RE, FXFALSE);

grDrawTriangle(&a, &b, &c);
```

¿NECESITAS

LA MEJOR REVISTA DE INFORMÁTICA

INFORMARTE,

LA MEJOR REVISTA PARA USUARIOS DE INTERNET

SUPER
NET
MAGAZINE
BANCA ELECTRÓNICA
YA ES POSIBLE REALIZAR LAS OPERACIONES BANCARIAS DESDE CASA
Analizamos las opciones con más éxito en nuestro país y su futuro
PROFESIONALES
La importancia de una agencia de publicidad en el futuro de la Red
GIF S ANIMADOS
Trucos para captar más visitas
PRENSA DIGITAL
La que no encontrara en papel
SEMANA SANTA
Planee sus vacaciones en la Red
ANALIZAMOS:
NetIII Central 2.0
MS Exchange Server 5.5
Smadia y Mail Server 1.2
Globelink Web Translator 1.3
Creative WebCam II

NAVEGAR,

LA MEJOR REVISTA DE JUEGOS

JUGAR,

LA MEJOR REVISTA DE PROGRAMACIÓN

PROGRAMAR?

TOWER COMMUNICATIONS
C/ Aragoneses, 7 28108 Alcobendas (Madrid)
Tlfno. (91) 661.42.11 Fax. (91) 661.43.86

Windows Scripting Host (III)

Adolfo Aladro (aaladro@arrakis.es)

A lo largo de esta serie de artículos hemos apreciado la gran versatilidad y potencia de los scripts que podemos realizar utilizando la tecnología *WSH*. En este nuevo capítulo nos adentraremos en más ejemplos a fin de afianzar los conocimientos adquiridos así como introducir nuevos elementos. El objetivo es que seamos capaces de programar todo tipo de scripts con total facilidad.

AUTOMATIZACIÓN

La inclusión de objetos predefinidos es una de las posibilidades más interesantes de los *scripts* desarrollados con *WSH*. En los artículos anteriores ya introducimos esta posibilidad y ahora vamos a profundizar un poco más en ella. Como sabemos existen objetos predefinidos dentro del propio sistema *Windows* que son compartidos entre las distintos entornos que los programadores utilizan para desarrollar sus aplicaciones.

El objetivo consiste en ser capaces de programar todo tipo de scripts

Así por ejemplo cada vez que un programador quiere desarrollar un programa en el que en algún momento se quiere que el usuario pueda seleccionar un archivo local utiliza el *Common Dialog*, que es un objeto predefinido que se encuentra ubicado en el archivo **COMDLG32.ocx**.

La utilización de este objeto será una tarea relativamente sencilla si trabajamos en un entorno visual ya que consistirá en pinchar en una tabla de herramientas y arrastrar el icono

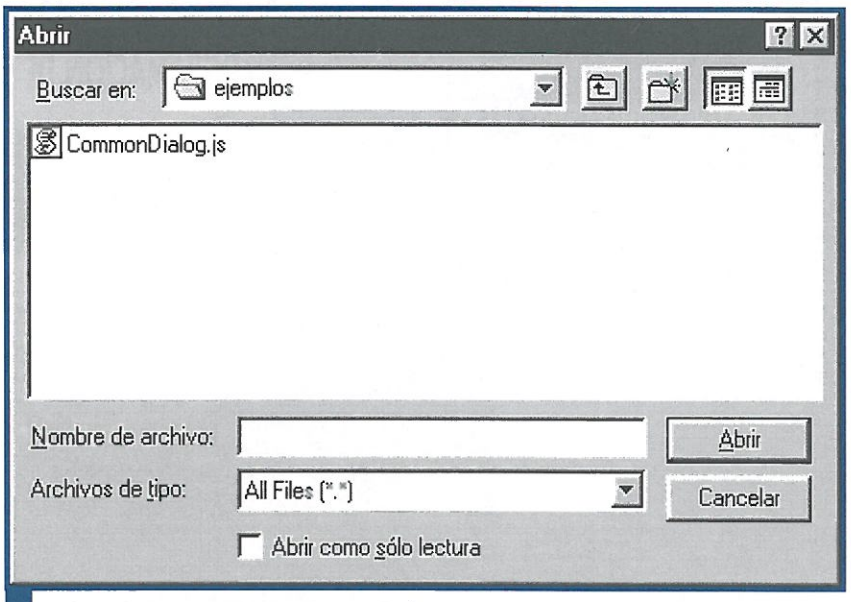


Figura 1. Aspecto que presenta Common Dialog.

correspondiente al formulario que representa nuestro programa.

Pero cuando queremos utilizar estos objetos desde un *script* no resulta fácil saber cómo debemos llamarlos o cuáles son sus propiedades y métodos. Para asistirnos en este tipo de tareas existen algunas herramientas de ayuda gracias a las que podemos conocer qué objetos se encuentran a nuestra disposición en el sistema para ser utilizados.

La inclusión de objetos predefinidos es una de las posibilidades más interesantes de los scripts

Asimismo podemos obtener toda la información sobre cómo llamar a esos objetos, cuáles son sus métodos y cuáles son sus propiedades, etc. De entre todas las utilidades que podemos encontrar para este fin vamos a destacar dos: *OLE/COM Object Viewer* y *VBA Companion*. La **Figura 2** y la **Figura 3** muestran respectivamente el aspecto que presentan ambas herramientas.

OLE/COM Object Viewer
http://cwashington.netreach.net/downloads/files/ovi386.zip
VBA Companion
ftp://ftp.apexsc.com/pub/demos/vba-ocx32.exe
Microsoft Office 97 Automation Help
http://support.microsoft.com/support/kb/articles/Q167/2/23.asp

Finalmente otra característica interesante es la que se deriva de la utilización de las aplicaciones *Microsoft Office* desde los *scripts WSH*. Para obtener más información al respecto es recomendable descargar de Internet el archivo de ayuda

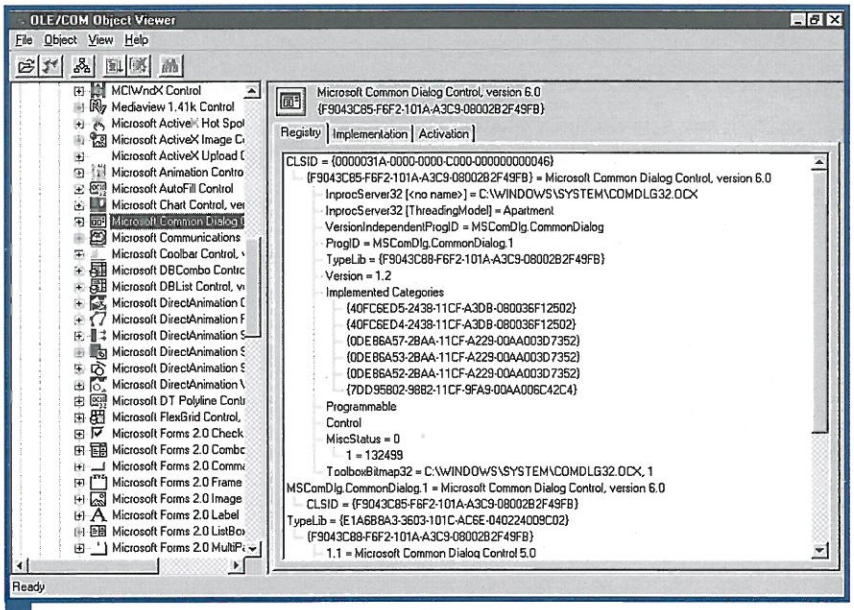


Figura 2. Aspecto que presenta la aplicación OLE/COM Object Viewer.

Auto97.hlp el cual proporciona información abundante la forma de automatizar estas aplicaciones.

EL CONTROL COMMON DIALOG

Pasemos ahora a ver unos cuantos ejemplos. El **Listado 1** muestra la forma de abrir un diálogo mediante el cual el usuario podrá seleccionar un archivo de su ordenador. Lo primero que tenemos que saber para utilizar este control es cómo crear una instancia de él. Para ello podemos hacer uso de la herramienta *OLE/COM Object Viewer*. Buscaremos dentro de la opción **Controls** y después seleccionaremos **Microsoft Common Dialog Control**. A continuación aparecerá gran cantidad de información sobre dicho control. De momento tendremos que fijarnos tan sólo en la línea *VersionIndependent ProgID* y anotar su valor. En el caso que nos ocupa será **MSComDlg.Common Dialog**. Con este dato ya sabremos cómo crear una instancia del objeto.

```
objDlg = new ActiveXObject
    ("MSComDlg.CommonDialog");
```

Una vez que tenemos una instancia del objeto podemos pasar a manipularlo utilizando sus métodos y propiedades. La herramienta *VBA Companion* proporciona una excelente ayuda en este aspec-

Listado 1. Script que muestra la utilización del objeto Common Dialog.

```
objDlg = new ActiveXObject("MSComDlg.CommonDialog");
objDlg.Filter = "All Files (*.*)|*.*";
objDlg.FilterIndex = 2;
objDlg.MaxFileSize = 260;
objDlg.CancelError = false;
objDlg.ShowOpen();
WScript.Echo(objDlg.Filename);
```

Listado 2. Script que lee un archivo seleccionado por el usuario y ordena alfabéticamente sus líneas.

```
objDlg = new ActiveXObject("MSComDlg.CommonDialog");
objDlg.Filter = "Ficheros de texto (*.txt)|*.txtlJScript (*.js)|*.js";
objDlg.FilterIndex = 2;
objDlg.MaxFileSize = 260;
objDlg.CancelError = false;
objDlg.ShowOpen();

if (objDlg.Filename != "") {
 fso = new ActiveXObject("Scripting.FileSystemObject");
 f = fso.GetFile(objDlg.Filename);
 ForReading = 1;
 TristateUseDefault = -2
 ts = f.OpenAsTextStream(ForReading, TristateUseDefault);
 array_lineas = new Array();
 i = 0;
 while (!ts.AtEndOfStream) {
  array_lineas[i] = ts.ReadLine( );
  i++;
 }
 ts.Close( );
 array_lineas.sort();
 resultado = '';
 for(i=0;i<array_lineas.length;i++) {
  resultado += array_lineas[i] + '\n';
 }
 WScript.Echo(resultado);
}
```

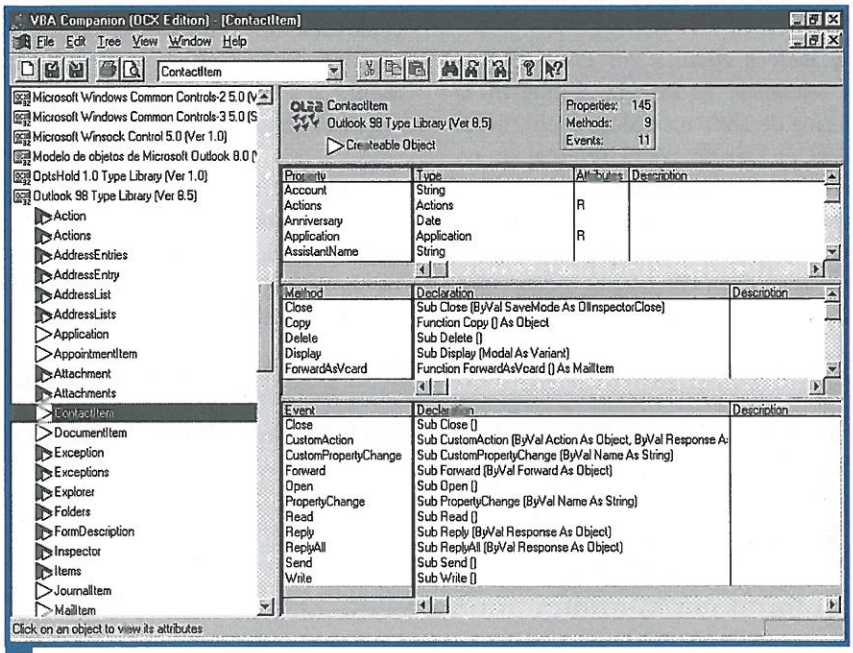


Figura 3. Aspecto que presenta la aplicación VBA Companion.

to. Basta con buscar, de la misma forma que lo hicimos anteriormente, el control correspondiente y la herramienta mostrará una relación de las propiedades y métodos del mismo. Por el momento sólo nos vamos a detener en unas cuantas propiedades y en el método gracias al cual se visualizará el diálogo.

- **Filter**

Esta propiedad permite fijar las opciones de filtrado de los archivos que muestra el control. Su valor será una cadena formada por elementos separados por el carácter |. Un filtro vendrá representado por dos elementos de esa cadena: el primero será la descripción del mismo y el segundo una cadena donde se expresará con comodines. Por ejemplo:

```
objDlg.Filter="JScript (*.js)
      |*.js|VBScript(*.vbs)|(*.vbs)";
```

crearía dos opciones de filtrado una con los archivos con extensión **JS**, que describimos como **"JScript (*.js)"** y otros los archivos con extensión **VBS** que describimos como **"VBScript (*.vbs)"**.

La herramienta VBA Companion va a proporcionar una excelente ayuda

- **CancelError**

Esta propiedad almacena un valor booleano. En caso de ser **true** hará que se genere un error cada vez que el usuario abandone el control sin seleccionar algún archivo. En el caso contrario, es decir, que el resulado sea **false** no se producirá ese error y simplemente podremos obtener una cadena vacía como nombre del fichero seleccionado.

- **Filename**

Esta propiedad guardará la ruta del archivo seleccionado.

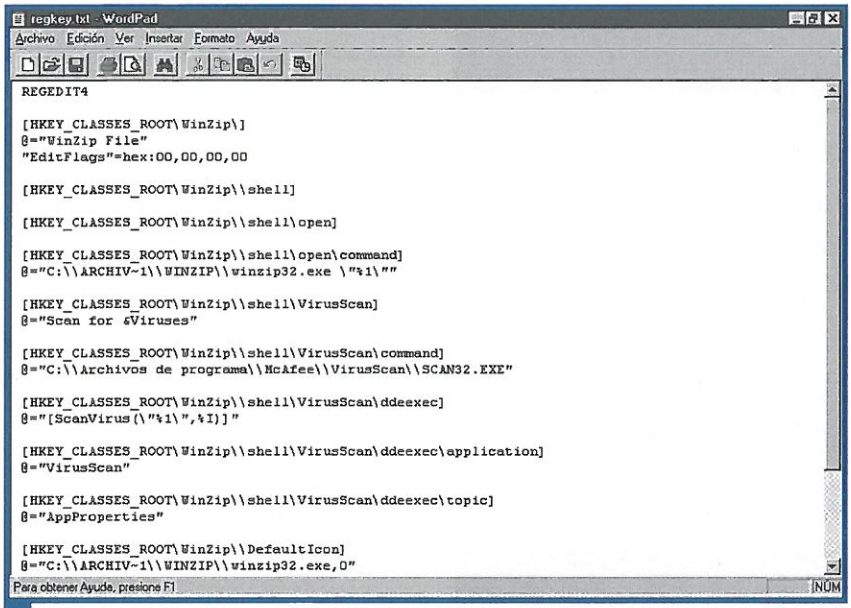


Figura 4. Se muestra el archivo regkey.txt con todas las ramas que cuelgan de la clave especificada en el registro de Windows.

- **ShowOpen()**

Llamaremos a este método una vez que hayamos fijado las propiedades del control y queramos que se muestre.

El **Listado 2** muestra un *script* algo más elaborado en el que también utilizamos el control *Common Dialog*. Tomamos un archivo de texto seleccionado por el usuario, lo leemos y mostramos una ventana con el contenido del archivo, con sus líneas ordenadas alfabéticamente.

La interfaz de control de medios se puede usar para manipular información multimedia

En este ejemplo cabe destacar la utilización del objeto *array* para ordenar elementos automáticamente gracias al método *sort()*. Abrimos el archivo como un objeto *TextStream* y utilizando el método *ReadLine()* vamos leyendo línea a línea todo el contenido del mismo.

Cada línea la almacenamos en un elemento del citado *array*. Finalmente lo ordenamos utilizando el método *sort()* que se define de la manera que indicamos a continuación:

```
arrayobj.sort(sortfunction)
```

donde *sortfunction* es un parámetro opcional que representa la función que se utiliza para la ordenación. Si se omite, se ordena alfabéticamente de manera ascendente. En el caso de que queramos otro tipo de ordenación crearemos una función que admita dos parámetros.

Si bWaitOnReturn es true el script esperará a que termine el programa antes de continuar con su ejecución

Ésta deberá devolver un valor negativo si el primer parámetro es menor que el segundo, cero si los dos parámetros son equivalentes y un valor positivo en otro caso.

LA INTERFAZ DE CONTROL DE MEDIOS (MCI)

La interfaz de control de medios es uno de los elementos que podemos usar en nuestras aplicaciones para manipular información multimedia.

Listado 3. Script que reproduce un archivo WAV utilizando la interfaz de control de medios (MCI).

```
mcicontrol = new ActiveXObject("MCI.MMcontrol");
mcicontrol.FileName = "test.wav";
mcicontrol.AutoEnable = 1;
mcicontrol.Command = "Open";
mcicontrol.Command = "Play";
WScript.Echo("Terminado")
```

Listado 4. Script que lee el registro de Windows para saber la ruta donde se encuentra la aplicación WinZip.

```
wshShell = new ActiveXObject("WScript.Shell");
clave = "HKCR\\WinZip\\shell\\open\\command\\";
WScript.Echo(wshShell.regRead(clave));
clave = "HKCR\\WinZip\\DefaultIcon\\";
WScript.Echo(wshShell.regRead(clave));
```

Listado 5. Script que lee todas las claves de una rama del registro de Windows.

```
wshShell = new ActiveXObject("WScript.Shell");
fso = new ActiveXObject("Scripting.FileSystemObject");

wshShell.Run("command /c if exist c:\\temp\\regkey.txt del c:\\regkey.txt", 0, true);
wshShell.Run('regedit /e c:\\temp\\regkey.txt "HKEY_CLASSES_ROOT\\WinZip\\"', 0,
true);

if (!fso.FileExists('c:\\temp\\regkey.txt')) {
  WScript.Echo('La clave no existe');
} else {
  wshShell.Run("wordpad.exe c:\\temp\\regkey.txt")
}
```

Listado 6. Script que crea un contacto en la carpeta contactos de su aplicación Outlook.

```
AppOutlook = new ActiveXObject("Outlook.Application");
OlContactItem = 2;
contacto = appOutlook.CreateItem(olContactItem);
contacto.FirstName = "Pepe";
contacto.LastName = "Pepa";
contacto.BusinessTelephoneNumber = "33-333-333";
contacto.Email1Address = "pepa@pepe.es";
contacto.Birthday = "22/04/89";
contacto.Save();
WScript.Echo("Terminado");
```

MCI representa un meta-lenguaje que permite "hablar" con los controladores de ciertos dispositivos multimedia para solicitarles información, ordenarles determinada tareas, etc.

El método Run() del objeto WScript.Shell sirve para ejecutar un programa en otro proceso

Cualquier control **(MCI32. ocx)** podemos utilizarlo en nuestros *scripts*. Tal y como ya hemos explicado anteriormente acudiremos a la aplicación *OLE/COM Object Viewer* para tratar de saber cómo podemos crear, de la mejor manera, una instancia de este objeto:

```
mcicontrol = new
    ActiveXObject("MCI.MMcontrol");
```

y posteriormente podremos profundizar en sus propiedades y métodos con la información que proporciona *VBA Companion*.

AÑADIENDO CONTACTOS CON OUTLOOK

Terminando con este amplio apartado de *Windows Scripting Host* dedicado a la automatización vamos a ver un ejemplo en el que manejamos la aplicación *Outlook* desde un *script* para añadir un contacto a la lista de contactos, para ello se puede consultar el **listado 5.**

En primer lugar creamos una instancia de la aplicación *Outlook* y posteriormente creamos un elemento, que en nuestro caso será un contacto. El tipo de elemento vendrá determinado por el valor de su parámetro, que en este caso será *olContactItem* (Estos valores los podemos obtener también utilizando la aplicación *VBA Companion*).

El contacto aparecerá automáticamente en la carpeta de contactos tal y como lo hemos escrito

Una vez que tenemos creada una referencia a un contacto, simplemente asignamos valores a cada una de sus propiedades y lo salvamos. El contacto aparecerá automáticamente en la carpeta de contactos tal y como lo hemos escrito **(Ver Figura 5).**

EL REGISTRO DE WINDOWS

En el capítulo primero de esta serie ya apuntamos la posibilidad de manejar el registro de *Windows* utilizando *Windows Scripting Host*:

- **RegDelete, RegRead**

Estos métodos toman el nombre de una entrada del registro y lo utilizan para borrar o leer su contenido respectivamente.

- **RegWrite**

El método *RegWrite* se utiliza para escribir entradas en el registro. Para poder realizar esto, hay que especifi-

car la ruta, el valor y el tipo de los datos que se van a escribir (*REG_SZ, REG_EXPAND_SZ, REG_DWORD* ó *REG_BINARY*).

El **Listado 4** contiene un ejemplo acerca de cómo podemos leer ciertas entradas del registro: la ruta de la aplicación *WinZip* y el archivo de donde el sistema toma el icono por defecto que se asigna a esa aplicación (Nótese que cuando creamos una cadena de texto en la que aparece el carácter '\' es necesario duplicarlo ya que por definición este carácter es el de escape).

Sin embargo estos dos métodos pueden resultar insuficientes a la hora de manejar el registro de *Windows* ya que permiten acceder a entradas concretas del registro pero no podemos navegar por él. Sin embargo se puede superar esta carencia ayudándonos de la aplicación *regedit* que se incluye dentro de *Windows*.

El método Run() cuenta con dos parámetros más que son opcionales

El **Listado 5** muestra un ejemplo: buscamos una determinada clave en el registro y volcamos en un archivos todas las ramas que surgen de esa clave. Este ejemplo no es sólo interesante por su utilidad sino porque además aparecen varios recursos de programación que hasta ahora no habíamos visto.

Sabemos que el método *Run()* del objeto *WScript.Shell* sirve para ejecutar un programa en otro proceso. Pues bien, podemos aprovechar la funcionalidad de este método para ejecutar comandos de nuestro viejo sistema *MS-DOS*. Esto es lo que hacemos la primera vez que invocamos al método *Run()* en el *script* del

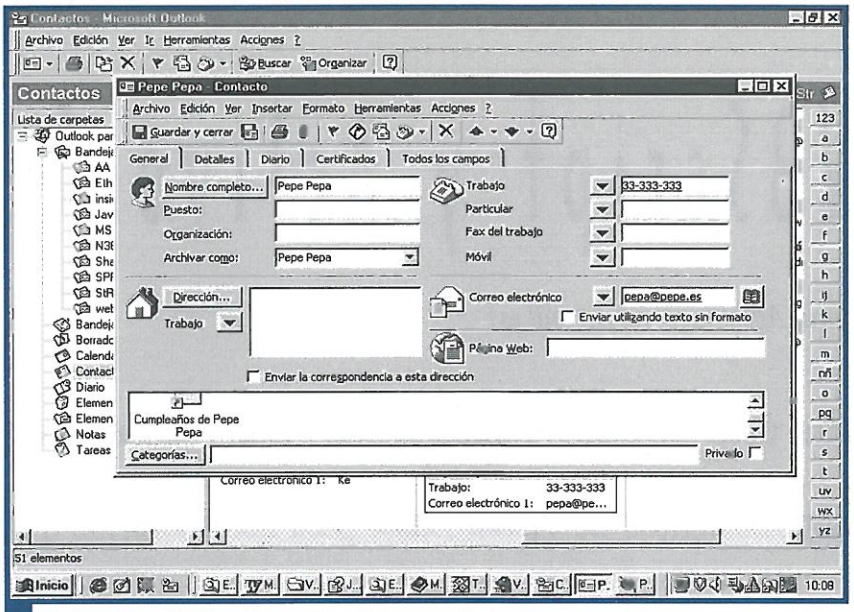


Figura 5. Entrada que se ha creado en la carpeta de contactos tras ejecutar nuestro script.

Listado 5. Llamamos al **COMMAND.com**, que es el programa que ejecuta los comandos de *MS-DOS*, para poder ver si existe el archivo **c:\temp\regedit.txt**. En caso afirmativo borramos el archivo utilizando el comando *DEL*. Esto mismo podríamos haberlo logrado utilizando *JavaScript* haciendo lo siguiente:

```
if (!fso.FileExists('c:\\temp\\regkey.txt')) {
 fso.DeleteFile('c:\\temp\\regkey.txt');
}
```

El método *Run()* cuenta con dos parámetros más (opcionales) además de la cadena donde se expresa el programa que queremos ejecutar. El primero de ellos al que se le llama *intWindowStype* es el que expresa el tipo de ventana *Windows* en el que se ejecutará el programa. El segundo se denomina *bWaitOnReturn*. Si este parámetro resulta ser **false**, o resulta no estar definido, el *script* no esperará a que el programa correspondiente termine, es decir, se lanzará un proceso paralelo al ejecutarse el método *Run()*. Si *bWaitOnReturn* es **true** el *script* esperará a que termine el programa antes de continuar con el resto de su ejecución.

La segunda vez que ejecutamos el método *Run()* en nuestro *script* es para llamar a la aplicación *regedit*. Lo que haremos será pasarle una clave que buscar y le pediremos que escriba todas las ramas que cuelguen de esa clave en un archivo de texto.

Finalmente comprobamos si el archivo existe, en cuyo caso volvemos al invocar al método *Run()* abriendo el archivo *regkey.txt* con el *wordpad*.

Con WSH tenemos la posibilidad de poder manejar el registro de Windows

La **Figura 4** muestra la aplicación *wordpad* con la información que hemos obtenido. Una vez que hemos conseguido tener creado este archivo de texto podríamos manipularlo de la manera que deseemos. Incluso podríamos llegar a pensar en poder realizar una página *Web* basada en *HTML* Dinámico, que haciendo uso de un *script* similar a éste, constituyera una interfaz para navegar por el registro de *Windows*.

VBScript y Active Server Pages

Jordi Agost (agost@eup.udl.es)

En este artículo vamos a ver cómo podemos aprovechar todos nuestros conocimientos de *Visual Basic* para incorporarlos a *Internet*. Empezaremos dando una visión general del lenguaje *VBScript* (subconjunto de *Visual Basic*) para terminar con la explicación de las páginas de servidor activo o *Active Server Pages*.

INTRODUCCIÓN AL VBSCRIPT

El lenguaje de programación *Visual Basic* posee algunos dialectos, entre los que cabe destacar *Visual Basic* para aplicaciones (*VBA*), diseñado principalmente para ser utilizado dentro de las aplicaciones de *Office*, para la programación de este tipo de aplicaciones. Otro dialecto es *VBScript*, que consta de un pequeño subconjunto del lenguaje original *Visual Basic* que puede ser interpretado por los siguientes programas: *Internet Explorer*, *Internet Information Server*, *Outlook* y el *host* de automatización de *Windows* (*WSH*). El código que escribimos con *VBScript* es interpretado y ejecutado por un motor de automatización, que no es más que una *DLL* llamada **VBSCRIPT.DLL** localizada en el directorio del sistema. Este subconjunto se instala en el mismo momento que las aplicaciones nombradas anteriormente.

Es necesario asegurarnos que tenemos una versión posterior a la 3.0 para realizar los ejemplos que veremos en este artículo. Esto lo podemos saber consultando las propiedades de la *DLL* desde el administrador de archivos.

Dicho lenguaje o motor de automatización lo podemos utilizar para ejecutar órdenes tal como lo realiza un archivo de proceso por lotes en *Windows*, para automatizar tareas (como los archivos *BAT* en *DOS*). También es posible emplearlo dentro de Internet, de tal forma que un usuario podrá dotar a las páginas *HTML* de una mayor funcionalidad (*VBScript* se puede incrustar dentro de una página *HTML* estándar). Bajo la perspectiva del servidor lo podemos utilizar junto con las *Active Server Pages* ó *ASP* para alterar las páginas *Web* antes de devolver su contenido, tema que veremos más adelante en este mismo artículo.

También podemos ejecutar *VBScript* en el escritorio o el navegador del usuario. Dentro de una página o un fichero *HTML* es posible incluir código *VBScript*. Por ejemplo si lo que queremos es incluir un mensaje lo podríamos hacer con el siguiente código:

```
<SCRIPT Language = "VBScript">
<!-
MsgBox "Hola Mundo"
—>
</SCRIPT>
<FONT SIZE = "+2"> Esto es una prueba de
    página Web
```

En el que podemos observar cómo estamos mezclando código *VBScript* con código *HTML*. Simplemente antes de utilizar código *VBScript* debemos indicárselo al navegador con la etiqueta <SCRIPT>.

EL LENGUAJE VBSCRIPT

Una de las principales diferencias entre *VB* y *VBScript* es que todas las variables de *VBScript*, sean del tipo que sean (globales, procedimientos, funciones, etc.) han de pertenecer al tipo *Variant*. Para ello disponemos de algunas funciones que nos pueden indicar cuál es

el tipo de variable almacenado en una variable *Variant*. Dichas funciones son *VarType* (que devuelve una constante con el tipo de la variable), *TypeName* (que devuelve una descripción del tipo de la variable) y funciones del tipo *Is()* que permiten comprobar tipos específicos (p. ej. *IsDate()* para las fechas).

Del mismo modo no se puede conectar un nombre de variable después de una instrucción *Next* perteneciente a un bucle *For*. Otra limitación se refiere a la sustitución de algunas funciones de *Visual Basic* por otras que realizan el mismo trabajo pero de una forma distinta para evitar dañar a la computadora host (la que ejecuta el código en ese momento). Así todo el sistema de archivos se ha visto modificado, y en su lugar el acceso a archivos se realiza a través de propiedades y métodos de ciertos objetos especiales. Los objetos especiales, empleados para trabajar con archivos, son los siguientes:

- **File**: representa un archivo de la máquina local en el que se ejecuta el código de VBScript.
- **Folder**: este objeto representa una carpeta de la máquina local.
- **Drive**: representa una cierta unidad de la máquina.
- **TextStream**: este objeto proporciona posibilidades secuenciales para la lectura o escritura.
- **FileSystemObject**: representa todo el sistema de archivos y sirve para referenciar todos los objetos enumerados en la tabla.

Pero aunque estemos limitados por la propia naturaleza del lenguaje, siempre puede sacarnos de un apuro la comunicación con otros objetos, como por ejemplo *DLLs ActiveX* y así aprovechar sus funcionalidades. Para hacerlo, primero deberíamos crear una instancia con el objeto con el que deseamos comunicarnos (recordemos que éste debe estar convenientemente registrado en el sistema). Por ejemplo si queremos una comunicación con *Microsoft Word* haríamos lo siguiente:

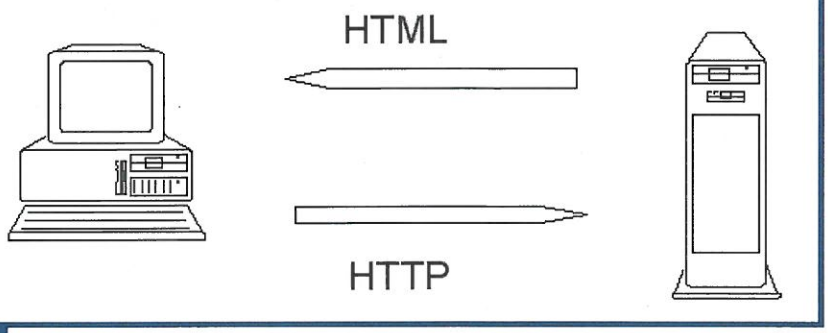


Figura 1. El servidor *Web* envía una página *HTML* al navegador como respuesta a la solicitud de información efectuada.

```
Set Obj = CreateObject("Word.Application")
```

Luego, como ya tenemos una variable que hace referencia al objeto, podremos en cualquier momento utilizar sus métodos y propiedades. Así para insertar un texto en *Word* haríamos lo siguiente:

```
Obj.Visible = True
Obj.Activate
Obj.Documents.Add
Obj.WordBasic.Insert "Hola mundo"
```

Donde Visible es una propiedad del objeto *Word*, al que se hace referencia con la variable *Obj*. Por otra parte, *Activate* es un método del objeto *Word* y así sucesivamente. Para finalizar deberíamos destruir la variable que apunta al objeto, ya que en caso contrario continuaría ocupando memoria de forma innecesaria.

```
Set Obj = Nothing
```

VBScript también se usa para mejorar la interactividad de las páginas *Web*. Así, lo podemos usar bajo el punto de vista del servidor, donde el código que insertemos será independiente del mismo, o bien lo podemos usar desde la perspectiva del cliente, donde tendremos la limitación de que el usuario debe tener *Microsoft Internet Explorer* instalado para su ejecución.

Al igual que en *Visual Basic*, en *VBScript* también se responde a eventos, colocando código en procedimientos de evento. En el siguiente ejemplo calculamos la suma de dos cuadros de texto de un sencillo formulario que hemos creado para la ocasión. Dichos elementos los creamos a través del código *HTML* siguiente:

```
<INPUT Type = "Text" Name="txtNum1"
    Value="0">
<INPUT Type = "Text" Name="txtNum2"
    Value="0">
```

Y dado que dichos elementos tienen un nombre, lo que vamos a hacer es escribir una rutina que cogiendo los elementos adecuados ejecute la suma correspondiente. Pare ello tan sólo realizamos la rutina tal y como la escribiríamos en *Visual Basic* (recordemos que todas las variables son de tipo *Variant*). Por tanto, la rutina quedaría así:

```
<HTML>

<SCRIPT Language = "VBScript">
<!—

Sub cmdCalculate_OnClick()
    Dim num1
    Dim num2
    Dim resul

    num1 = Cint(txtNum1.value)
    num2 = CInt(txtNum2.value)
    resul = Suma(num1,num2)

    MsgBox "La suma es " & resul
End Sub
—>
</SCRIPT>
```

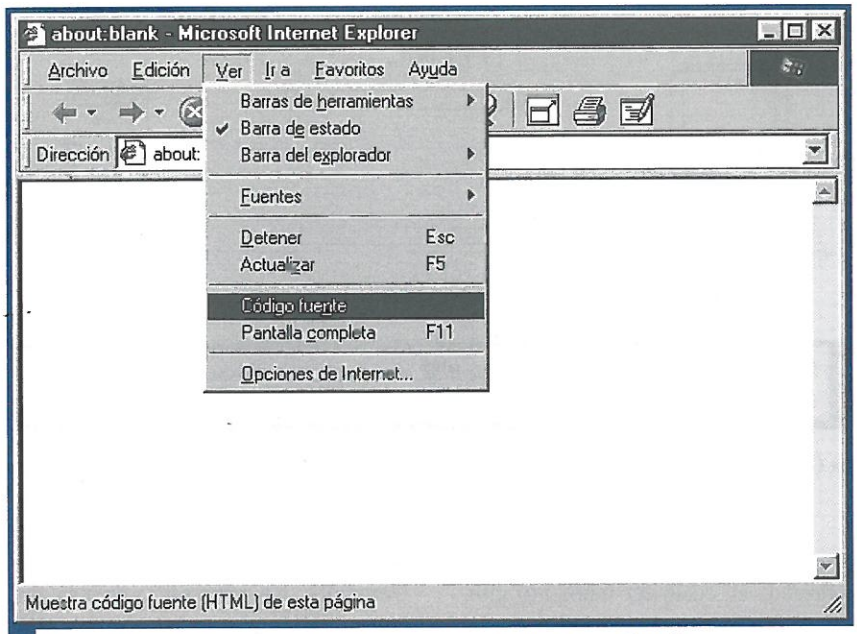


Figura 2. A través de esta opción podemos examinar el código fuente (*HTML*) que nos envía el servidor.

```
<BODY>
<BR>

Prueba de suma de dos
    números<BR><BR><BR>
Introduzca el primer número:<INPUT
    Type="Text" Name="txtNum1"
    Value="0"><BR><BR>
Introduzca el segundo número:<INPUT
    Type="Text" Name="txtNum2"
    Value="0"><BR>
<BR><BR>
<INPUT Type = "Button"
    Name="cmdCalculate" Value= "Calcular
    suma">

</BODY>
</HTML>
```

FORMULARIOS

Una característica del *HTML*, y que posteriormente será útil para trabajar con páginas *ASP* en el servidor, son los formularios. Estos son utilizados para devolver información al servidor *Web*. Observemos un típico ejemplo de un formulario con un cuadro de texto:

```
<FORM ACTION=buscar.asp
    METHOD=POST>

Introduzca la compañía:
<INPUT TYPE=TEXT NAME=txtBuscar>
<INPUT TYPE=SUBMIT VALUE="Buscar">
</FORM>
```

El botón *Submit* se utiliza para indicar al navegador que debe enviar todos los campos, nombrados con la etiqueta *<INPUT>* y que están entre las etiquetas *<FORM>* al servidor *Web*. Dicho servidor podrá acceder a cada uno de estos elementos y colocarlos en una base de datos.

PÁGINAS DE SERVIDOR ACTIVO (ACTIVE SERVER PAGES)

Mientras navegamos por *Internet* y nos vamos desplazando por sus innumerables páginas, podemos pensar cómo nos muestran el contenido dinámico, es decir, cómo procesa una determinada petición de información nuestra, o cuál es el proceso que se efectúa para mostrar algún tipo de información en tiempo real.

La forma de llevar a cabo estos procedimientos consiste en realizar un programa que se ejecute en el servidor *Web* que actúe leyendo las entradas que son enviadas por el usuario que navega y generando de forma dinámica una página *Web* con el resultado apropiado. Como contrapartida al mundo *UNIX*, donde dichos programas normalmente son scripts en *CGI* o bien programas en *C*, *Microsoft* ha creado las *Active Server Pages*, que son un componente del servidor *Web* mediante los que podremos desarrollar nuestros propios *scripts*.

El lenguaje utilizado para crear las páginas *Web* es el *HTML*, y dicho lenguaje posee un contenido estático, es decir el contenido de una página creada con *HTML* no cambia por sí mismo, siempre es el mismo.

Para corregir esto, la industria del software ha creado los *applets* Java, los scripts para el cliente, el *HTML* dinámico, controles *ActiveX*, etc. La diferencia con respecto a *ASP* radica en que este último resulta una mejora para el servidor, ya que todo el código se ejecuta en él, y tan solo devuelve *HTML* estándar que puede ser visualizable en todos los servidores. Es decir, el *ASP* tan sólo incrustará código de automatización del lado del servidor en páginas *Web*.

Si estamos desarrollando una solución para una intranet ó *Internet* y queremos evitar temas como la descarga de *applets* por el lado de cliente y además deseamos que nuestra aplicación sea visualizable en cualquier navegador, entonces *ASP* será una solución ideal.

Aunque si lo que deseamos es poder usar las páginas *ASP* entonces

también nos veremos obligados a utilizar el servidor de información de *Web* de *Microsoft* (*Internet Information Server*). Dicho producto se encuentra en dos versiones, una para *Windows 95* (es una versión reducida de su hermano mayor) y otra para *NT*. *Internet Information Server* (*IIS*) identificará una página como servidor activo por su extensión (*.ASP*).

CREACIÓN DE ARCHIVOS ASP

Si lo que queremos es crear archivos *ASP*, únicamente necesitamos un pequeño editor de textos, como por ejemplo el Bloc de notas para crear en él un archivo con la extensión *ASP*. Aunque, si lo deseamos también podemos optar por una herramienta más avanzada como *Visual Interdev*.

Nuestro archivo *ASP* podrá estar formado por una mezcla de código y de *HTML*, aunque también es posible que una página contenga uno de los dos elementos citados. Si tenemos una dirección de *Internet* como por ejemplo:

```
http://www.holamundo.com/holamundo.htm
```

Dicha página representará a una página *Web*. En cambio si tenemos una dirección de una página activa:

```
http://www.holamundo.com/holamundo.asp
```

dicha página puede dar lugar a multitud de páginas *Web*, ya que lo que realmente sucederáes que en el servidor se ejecutará un cierto programa o *script* (realizado en *VBScript*) y en ese momento se construirá dinámicamente la página que se devolverá al navegador del usuario. Por ejemplo, si en el servidor introducimos el siguiente código en un editor (con la extensión *ASP*):

```
<HTML>
<BODY>
La fecha actual es
<%
    Dim sTime
    STime = Now
    Response.Write(sTime)
%>

</HTML>
</BODY>
```

y un usuario consulta la página que acabamos de crear recibirá en su navegador un mensaje con la fecha y la hora actual. Y si va pulsando el botón Actualizar, la hora y la fecha aparecerán actualizadas. Esto sucederá porque en cada petición de actualización el servidor creará una página distinta para el usuario, que en realidad no recibirá ninguna página *ASP*, sino tan sólo una página *HTML* estándar. Para saber exactamente lo que recibimos podemos examinar el código fuente del explorador, a través de los menús **Ver->Código Fuente**, tal y como se muestra en la figura 2.

```
<HTML>
<BODY>
La fecha actual es 01/01/1999 22:30
</HTML>
</BODY>
```

Podemos ver un esquema del funcionamiento en la figura adjunta 3.

Cuando el servidor está procesando un determinado archivo *ASP*, lógicamente sólo ejecutará el código que le corresponde, es decir, aquel código que esté marcado como *script* del lado del servidor. Para marcar este código (que deberá ejecutar el servidor) lo podemos hacer de dos formas distintas: por una parte utilizar la etiqueta del lenguaje *HTML* <*SCRIPT*> con la opción *RUNAT="SERVER"* o bien utilizando el prefijo <% antes del código y el sufijo %> después de este código. De la primera de estas manera nuestro código quedaría de la siguiente forma:

```
<SCRIPT LANGUAGE="VBSCRIPT"
    RUNAT="SERVER">
<!— Aquí incluiríamos el código que no interesase—>
</SCRIPT>
```

y del segundo modo:

```
<%
<!— Aquí incluiríamos el código que no interesase—>
%>
```

Contrariamente al primer ejemplo, en este último no es necesario especificar un lenguaje de automatización, porque se supone que el lenguaje predeterminado de los archivos *ASP* es *Visual Basic Script*. Aunque si deseamos variar este dato predeterminado, tan sólo deberíamos poner al principio de los archivos *ASP* lo siguiente:

```
<% LANGUAGE="VBSCRIPTT" %>
```

Es importante hacer resaltar que dicha instrucción debe aparecer (al igual que la orden *Option Explicit* de *VBScript*) al principio del archivo, delante de la etiqueta *HTML*.

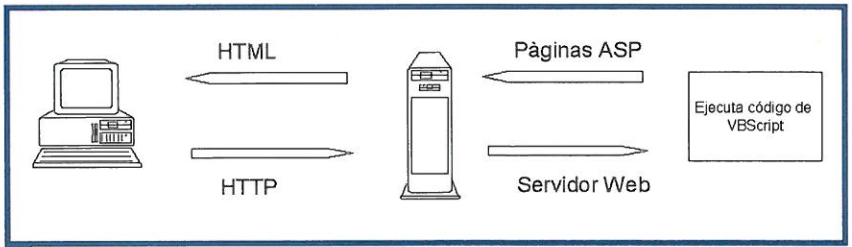


Figura 3. Las páginas activas de servidor son código de automatización ejecutado en el servidor. Este sólo devuelve páginas *HTML* estándares.

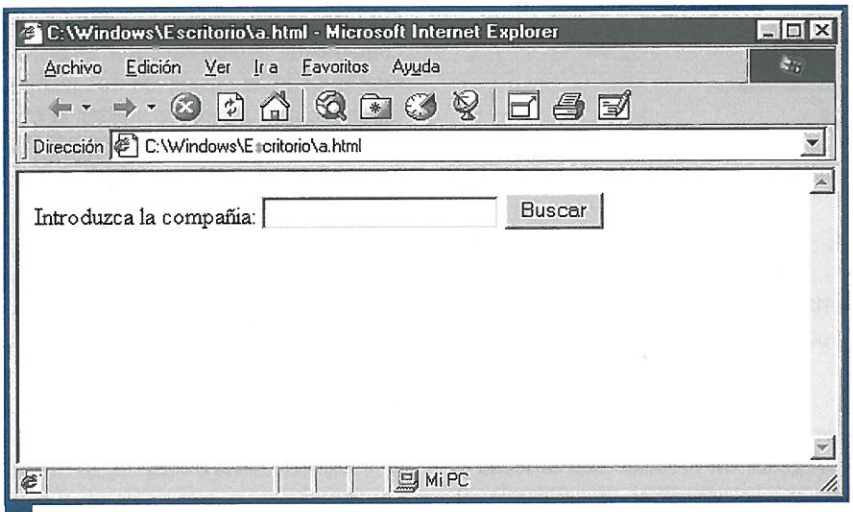


Figura 4. Muestra de la página *HTML* con el formulario que hemos creado.

LOS OBJETOS ASP

Las páginas activas de servidor poseen en su interior algunos objetos integrados, que a su vez son accesibles desde código de automatización. Los principales objetos disponibles para las páginas *ASP* son los siguientes:

- *Session*
- *Response*
- *Request*
- *Server*

El objeto *Session* lo utilizamos para crear lo que se denomina variables de sesión. Dichas variables son declaradas y se accede a ellas del mismo modo que lo haríamos con una colección. Normalmente se utiliza para retener la información que hay entre páginas. Si una información que ha sido enviada por el usuario a una página *ASP* desea ser guardada, entonces la página *ASP* incluirá dicho valor en una variable de sesión y pasará a estar disponible para otras páginas *ASP*. Si lo que queremos es crear una variable de este tipo, simplemente le asignamos un nombre y un valor:

```
Session("Estado") = "Abierto"
```

Donde *Estado* es el nombre y **Abierto** el valor. Una de las aplicaciones más habituales de las variables de sesión es evitar que los usuarios entren en una determinada dirección de *Internet* (o un determinado *URL*) saltándose la primera pantalla o la protección inicial. Si existe un archivo, un fichero *ASP*, que deseemos que posea características de seguridad simplemente agregaremos una instrucción *If* al inicio del mismo para verificar si se ha establecido una variable de sesión o no. Así podríamos incluir el siguiente código:

```
<%
    If Session("NombreUsuario")="" Then
      Response.Write "NO EXISTE UNA
    CONEXIÓN<BR>"
      Response.End
    End If

    Response.Write ("Hola " &
    Session("NombreUsuario"))
%>
```

En la rutina anterior simplemente examinamos si la variable de sesión *NombreUsuario* ha sido establecida o nó. En caso negativo terminamos la respuesta del servidor *Web* con el objeto *Response*. Vamos a ver a continuación dicho objeto, pero antes es preciso advertir que la protección propuesta anteriormente tan sólo se establece a nivel de aplicación y en ningún momento es establecida a nivel de red. Es decir, cualquier pirata informático podría coger nuestra línea y leer los datos que estamos enviando. Si quisiéramos evitar esto deberíamos realizar una conexión segura mediante el protocolo *HTTPS* (*Hypertext Transfer Protocol with Privacy* establece una conexión *HTTP* que usa encriptación del tipo *SSL*: *Segure Sockets Layer*).

Veamos ahora más detalladamente el objeto *Response* nombrado anteriormente. Este objeto se usa para enviar *HTML* o cualquier otro texto de vuelta al navegador. Con el método *Write* le enviamos al navegador lo que deseamos, tal y como hemos hecho en la rutina de seguridad anterior:

```
Response.Write ("Hola " &
    Session("NombreUsuario"))
```

También podríamos hacer que el servidor devolviera un determinado código (por ejemplo código hecho en *VBScript*) para que éste fuese interpretado por el explorador (recordemos que tan sólo *Internet Explorer* de *Microsoft* puede interpretar *VBScript*), tal y como se puede apreciar en el siguiente ejemplo:

```
<%
Response.Write ("<SCRIPT
    LANGUAGE=VBScript>" & Chr(13) &
    Chr(10))
Response.Write ("MsgBox " & Chr(34) & "Hola
    mundo" & Chr(34) & Chr(13) & Chr(10))
Response.Write ("</SCRIPT> & Chr(13) &
    Chr(10))
%>
```

el cual será recibido por el usuario en su navegador de la siguiente forma:

```
<SCRIPT LANGUAGE=VBScript>
MsgBox "Hola mundo"
</SCRIPT>
```

Recordemos que la combinación del carácter **13** y **10** producen un retorno de carro, mientras que **34** equivale a las comillas dobles (").

El objeto *Request* permite que el código *ASP* acceda a todo lo que se refiere una solicitud que viene del navegador. Podremos recibir información de formularios *HTML*, con la colección *Form* del objeto *Request*. Dicho objeto también posee la colección *ServerVariables*, que permite acceder a mucha información del objeto *Request* y del servidor por extensión.

El objeto *Server* sirve principalmente para crear objetos en el servidor *Web*, de forma que nuestra aplicación de páginas activas (o aplicación *ASP*) pueda acceder posteriormente a ellos. También podremos crear objetos a partir de clases que se pueden compilar en una *DLL ActiveX*. Posteriormente en el apartado de bases de datos veremos un ejemplo.

ACCESO A BASES DE DATOS CON ASP

Con *ASP* podemos utilizar objetos de tipo *ADO* (*ActiveX Data Objects*) dentro del código de *VBScript* para acceder a bases de datos. En primer lugar será necesario identificar un origen de datos. Supongamos que lo que queremos es abrir una página *HTML* y que en un momento dado el usuario pueda consultar los registros de una determinada base de datos ubicada en el servidor simplemente pulsando una botón. Supongamos que tenemos una base de datos relacional llamada **PRODUCT** en la cual hay una tabla llamada **PRODUCTOS** y dentro de ella un campo denominado **Compañía**. Dicha base de datos almacenará una lista de los productos de software disponibles en una hipotética tienda, y queremos que el usuario pueda, por ejemplo, consultar todos los productos disponibles de la compañía *Microsoft*.

Para ello lo primero que deberemos hacer es crear una página *HTML* estándar que envíe un determinado formulario a una página *ASP* (creada por nosotros y alojada en el servidor), la cual será la que realice la búsqueda. Aquí vemos el listado de la página *HTML* que muestra el formulario:

```
<HTML><BODY>
<FORM ACTION=buscar.asp
      METHOD=POST>

Introduzca la compañia:
<INPUT TYPE=TEXT NAME=txtBuscar>
<INPUT TYPE=SUBMIT VALUE="Buscar">
</FORM>
</BODY></HTML>
```

y en el código siguiente observamos el fichero **buscar.asp**.

Lo que hacemos en primer lugar es recuperar el valor que el usuario desea buscar del campo de formulario *txtBuscar*, y miramos si la cadena está vacía (sería el caso en que el usuario no hubiese escrito nada en el cuadro de texto). Posteriormente, establecemos una conexión con un objeto externo del tipo **Objeto de datos ActiveX** (*ADO - ActiveX Data Object*). Seguidamente establecemos como origen de datos nuestra base de datos (**conex.Open ="DSN=PRODUCT"**). Después construimos la cadena que contendrá nuestra consulta *SQL* y acto seguido la ejecutamos por medio de *ADO* para obtener un recordset.

Empezamos una tabla con código *HTML* y luego, dentro del código de *VBScript* nos movemos al principio del recordset que nos ha devuelto *ADO*, recorriéndolo hasta el final, añadiendo cada vez más elementos a la tabla. Como punto final cerramos las variables utilizada y liberamos memoria.

```
<HTML><BODY>

<%
Dim conex
Dim rset
Dim strsql
Dim strBuscar

strBuscar = Request.Form("txtBuscar")

If strBuscar="" Then
      Response.Write "Cadena Vacia")
      Response.End
End If

Set conex =
    · Server.CreateObject("ADODB.Connectio
      n")
conex.Open ="DSN=PRODUCT"

strsql = "SELECT * FROM Productos WHERE
      Compañía='" & strBuscar & "'" &
      ORDER BY PRODUCTO"
Set rset = conex.Execute(strsql)
%>

<TABLE BORDER=1>
<TR>Productos por compañías</TR>

<%
rset.MoveFirst
While Not rset.EOF
      Response.Write ("<TR><TD>")
      Response.Write rset.Fields("Programa")
      Response.Write ("</TD></TR>")
      rset.MoveNext
Wend

rset.Close
conex.Close
Set rset = Nothing
Set connex = Nothing
%>

</TABLE>
</BODY></HTML>
```

Hasta aquí hemos visto una primera aproximación a la programación de *Internet* aprovechando nuestros conocimientos de *Visual Basic*. A partir de este punto, ya tenemos la base suficiente como para empezar a entender un nuevo tipo de aplicaciones que viene incorporado en *Visual Basic 6.0*: las aplicaciones *IIS*. En sucesivos artículos veremos que son las *DLL ActiveX* que funcionan en un servidor y devuelven *HTML* al navegador del usuario. Algo parecido a las *ASP* pero enfocado a la creación de aplicaciones basadas en *Web*, más que a la creación de scripts.

Tratamiento de imágenes (I)

Javier Sanz Alamillo (jsanza@teleline.es)

En las primeras versiones de *Java* se disponía de las clases java.awt y java.awt.image para realizar un manejo de las imágenes algo reducido pero sencillo y eficaz. La aparición del *Java 2D API* ha ampliado estas posibilidades y ahora se puede realizar un tratamiento de imágenes mucho más potente, manteniéndose la sencillez de utilización de los comienzos.

En las primeras versiones de Java, como en la 1.0, el *AWT* ofrecía una serie de posibilidades que no pasaban de la utilización de imágenes de la misma forma que se usaban al crear las páginas *HTML*. No se podían realizar una serie de tareas algo más complejas sobre los gráficos, el texto o las imágenes, tan comunes en cualquier otro lenguaje de programación. La clase *Graphics* básicamente permitía dibujar rectángulos, polígonos, mostrar textos y visualizar, ampliar o reducir automáticamente imágenes.

INTRODUCCIÓN

Con la aparición del *Java 2D API* en el entorno de desarrollo de la versión de *Java 1.2* se amplían las posibilidades de uso. Con esta nueva *API* se pueden dibujar superficies, curvas, arcos, etc., así como transformaciones lineales sobre imágenes, incluyendo el escalado, la rotación, cambio de origen, etc. Se definen regiones de *clipping*, se permite la mezcla de imágenes (composición) mediante el uso de *AlphaComposite*, los textos se pueden girar fácilmente, ampliar, reescalar, etc.

Las primeras versiones de Java ofrecían importantes limitaciones en el tratamiento multimedia

Aunque no lo parezca, todas estas nuevas posibilidades, al igual que las que ofrece por ejemplo *Java Sound Engine*, se han ido incluyendo en un gran conjunto, una gran *API*, que ha conseguido que con Java se puedan desarrollar aplicaciones multimedia de una forma estable y robusta, todo ello sin perder la compatibilidad con versiones anteriores del *AWT* y con independencia de la plataforma, algo verdaderamente importante e interesante.

RESEÑA HISTÓRICA

Las primeras versiones de *Java* eran algo limitadas para el tratamiento multimedia, ya que ofrecían un mínimo soporte *API*, por lo que por ejemplo, el sonido en los *applets* se reducía a reproducir ficheros con extensión *AU*, existiendo además una *API* no oficial aparte para reproducir sonido en las aplicaciones. Esta circunstancia seguía ocurriendo incluso en la versión *JDK 1.1.7*. Para completar el conjunto de carencias, no se encontraba nada sobre el tratamiento de vídeo, menos aún sobre *stream* de *audio* o soporte para los gráficos en 2D ó 3D, exceptuando las rutinas básicas de dibujo de figuras, como líneas, rectángulos,

etc. Por lo tanto, podemos concluir que el soporte multimedia en *Java* durante sus inicios fue nulo.

Para remediar esta situación, *Sun* y sus *partners* se pusieron manos a la obra para ofrecer una "*Media API*" que permitiera el tratamiento multimedia en todos sus aspectos y de este trabajo surge la *API* de multimedia y de comunicaciones, conocida como *Media and Communication API*. En la siguiente tabla se muestra el avance en el desarrollo de las *Media APIs* de Java.

La utilidades más interesantes de las *APIs* presentadas al inicio de 1998 se resumen en:

- *API Java 2D*, permite el tratamiento de gráficos e imágenes.
- *Java 3D*, permite la realización de trabajos *3D*.
- *Java Sound*, permite instrumentar sonido *MIDI* y procesado de ficheros de sonido, como *WAV*.
- *Java Speech*, permite trabajar con el reconocimiento de voz.

En Marzo de 1998 se presentó en *JavaOne* otra serie de *Media APIs* más o menos definidas que son:

- *Java Advanced Imaging*, que permite el tratamiento de imágenes con unas características nuevas y técnicamente mejoradas.
- *Java Animation*, para la creación de animaciones *2D*.
- *Java Share Data Toolkit*, para el tratamiento de datos multimedia compartidos de forma concurrente, multiusuario, distribuida, etc.
- *Java Media FrameWork*, que permite un tratamiento en todos los aspectos del vídeo, siendo totalmente multiplataforma.

Sun consigue ofrecer, unos meses después, una *Media API* que posibilita desarrollar un entorno multimedia en el que se dispone de sonido, imágenes *2D*, modelos *3D*, vídeo, telefonía, animaciones, reconocimiento de voz, etc.

Tabla 1. Desarrollo de las Media APIs.

Versión Java	Disponible	Soporte
1995 – v1.0, release inicial, en todos los navegadores, (muy difundido).	En todos los SO's	AWT básico, audio limitado, Nada de vídeo ni 2D o 3D
1997 – v1.1	En algunos browsers y SO's, con ciertos problemas.	Sin cambios
1998 – v1.2	Aún en versión beta.	Java 2D, Java 3D, sonido, Reconocimiento de habla, API de vídeo....

El estado de la *Media and Communication APIs* se describe en la tabla 2.

AMPLIACIÓN DE LAS MEDIA APIS ACTUALES

Gracias a las *Media APIs* se nos ofrece por ejemplo, una ampliación del *AWT* actual, pero esto no está exento de problemas. Esto supone un verdadero cambio en todas las plataformas disponibles para *Java* y para que esto se cumpla, se requiere un periodo de adaptación importante. Además, una parte de las *APIs* están implementadas parcialmente y en algunos casos, sólo existen referencias escritas y no se encuentran implementaciones reales.

También existen otros problemas añadidos al crecimiento del *AWT*, y ya más ampliamente, de las *Medias APIs*, el entorno multimedia para *Java*. Las empresas competidoras frente a *Sun* en desarrollos multimedia, como por ejemplo *Apple* o *Microsoft*, no ven con buenos ojos el desarrollo de las mismas, puesto que algunos de sus productos son competencia directa.

En el caso de *Apple*, que si por algo destaca es por su importancia en el campo multimedia, únicamente ha empezado a apoyar este crecimiento mediante el desarrollo de su *QuickTime* para *Java*, que a su vez, de algún modo, es competencia directa de la *API Java Media Framework*. En el caso de *Microsoft* no es que apoye la iniciativa, sino que directamente no ofrece ningún tipo de apoyo a su creación, y se ha alineado con otras empresas, como *VRML Consortium* y *Silicon Graphics* para crear un frente común de "oposición" a este gran *API* para desarrollos multimedia.

Con la aparición de la API Java 2D se amplían las posibilidades multimedia de Java

Ya incluso por parte de *Sun* se muestran sus flaquezas. Si se utilizan las *Media APIs* las aplicaciones desarrolladas no pasarán el test de **100% Pure Java**, por lo que muchos vendedores de software y desarrolladores están en una situación de espera, atentos a la forma en que se debe implementar y cuándo se podrá realizar.

Para completar todas estas circunstancias, entramos en el mundo de los navegadores y sus batallas. *Netscape* y *Microsoft* afirman que sus navegadores integran totalmente las *Media APIs*, ya que ninguna puede quedarse fuera de la línea de desarrollo que impone *Sun* para *Java*. ¿Se conseguirá un completo y mul-

Tabla 2. Estado de la Media and Communication APIs.

Java Media API	Tipo de API	Versión API	Existen FAQ
Java 2D	Core 1.2	1.0 Final	Sí
Java 3D	Extensión.	1.1	Sí
Java Media FrameWork	Extensión	2.0	Sí
Java Sound	El engine, Core 1.2		
El API, extensión	No definida	Sí	
Java Speech	Extensión	1.0	Sí
Java Telephony	Extensión	1.2	Sí
Java Advanced Imaging	Extensión	2.0 beta	Sí
Java Share Data Toolkit	Extensión	1.4	Sí

tiplataforma *Java Media* gracias a la guerra de los navegadores?. Aún está por ver, aunque de todos modos, si algo ha cambiado, es que *Microsoft* ha tenido que adaptar toda la interfaz multimedia de su navegador *Explorer* para que sea del estilo *Java Media Compatible* ya que *Netscape* se ha volcado totalmente en el desarrollo de las *Media APIs*.

MANTENIMIENTO DE LA COMPATIBILIDAD E INDEPENDENCIA DE PLATAFORMA

El *API Java 2D* mantiene la compatibilidad con el *JDK 1.1*, por lo que los desarrollos con *Java 2D* mantienen la independencia de plataforma. Para garantizar esta compatibilidad, la funcionalidad de las clases e interfaces para el tratamiento de gráficos e imágenes se ha mantenido. Las características existentes no han sido eliminadas y no se ha alterado ningún paquete de *AWT* anterior. El *Java 2D* permite toda la funcionalidad del *AWT* pero añade nuevos métodos a las clases existentes, extendiendo las mismas y añadiendo nuevas clases e interfaces.

Por ejemplo, la mayoría de las características del *2D API* se han construido a partir de *Graphics2D*, que es un contexto gráfico ampliado. Para permitir esta ampliación, se ha derivado la clase *Graphics2D* de *Graphics*, esta última clase existente en el *JDK 1.1*.

El uso de los contextos gráficos no cambia, puesto que el *AWT* sigue manteniendo todos los métodos tradicionales de manejo de contextos gráficos:

- *paint*
- *paintAll*
- *update*
- *print*
- *printAll*
- *getGraphics*

Un *applet* desarrollado con el *JDK 1.1* interpreta el contexto gráfico que le es pasado como una instancia de *Graphics*. Para garantizar el uso de las nuevas características implementadas en *Graphics2D* un *applet* que utilice *Java 2D* debe únicamente realizar un *casting* del contexto gráfico *Graphics* a *Graphics2D*, por ejemplo:

```
public void paint (Graphics g)
{
  Graphics2D micg = (Graphics2D) g ;
}
```

En los casos que en los que se presenta una herencia de clases, *Java 2D API* utiliza dos formas de generalizar esta situación:

- Una o más clases base son añadidas en la jerarquía de clases y estas clases son utilizadas como clases bases para las nuevas, que implementan nuevos métodos y datos.
- Añadir nuevas interfaces de forma que son utilizadas para añadir nuevos métodos abstractos a las clases que heredan.

Por ejemplo, la *API Java 2D* generaliza el uso de la clase *Rectangle* del *AWT* aplicando las dos formas anteriores. La jerarquía de las clases sería ahora la siguiente:

```
java.lang.Object
  |
  |- java.awt.geom.RectangularShape
  |
  |- java.awt.geom.Rectangle2D
  |
  |- java.awt.Rectangle
```

En el *JDK 1.1* la clase *Rectangle* simplemente es derivada de *Object*. Con *Java 2D* se ha creado una nueva clase *Rectangle2D* que implementa *Shape* y *Serializable*. Se trata de dos clases base que se han añadido a la jerarquía de *Rectangle*: *RectangularShape* y *Rectangle2D*. Pese a ello, los *applets* escritos usando *JDK 1.1* no necesitarán

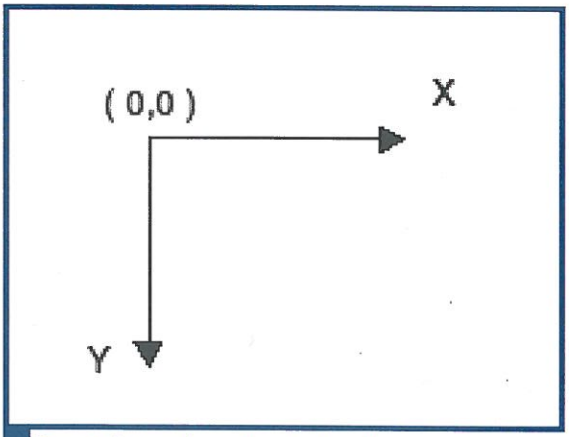


Figura 1. Sistema de coordenadas User Space.

¡NO TE LO PIERDAS!

2495 ptas. IVA incluido

Ahorra horas de busqueda en Internet

Especial énfasis en Webs en español

ABETO editorial

C/ Aragoneses, 7 – 28108 Pol. Ind. Alcobendas (MADRID)
Tel.: 91 661 42 11* – Fax: 91 661 43 86
http://www.abetoed.es

saber de la existencia de estas clases base y de las implementaciones de interfaces, puesto que los métodos y miembros de *Rectangle* de anteriores versiones aún se mantienen.

También, el *API Java 2D* añade nuevas clases e interfaces que van en paralelo a la jerarquía del *AWT* anterior. Estas clases e interfaces no derivan ni generalizan clases existentes, sino que son completamente nuevas y diferentes. Engloban características que no pueden ser integradas con versiones anteriores del *AWT*. Por ejemplo, el *API Java 2D* implementa nuevas clases *Shape* como son *Arc2D*, *CubicCurve2D*, etc., que se han añadido a la nueva *API* pero no pueden ser incluidas en la anterior jerarquía de clases porque aún se mantienen la *drawArc* y *fillArc*, soportada por la clase *Graphics*.

Las Media and Communication API's permiten trabajar con vídeo, audio, sonido, imagenes 2D, modelos, 3D, etc

La independencia de plataforma para las aplicaciones que usan *Java 2D API* se debe a que no se preocupan de asuntos como qué resolución se está utilizando, de cuántos colores se dispone, qué modelo de paleta se va a utilizar o qué tipo de dispositivo se empleará, ya sea una impresora o un monitor. Nada hace que la API Java 2D asuma un formato particular de fichero de imágenes o gráfico. Siempre hay que recordar que la verdadera independencia de plataforma se aplica cuando todo lo que se va a tratar se encuentra dentro de lo soportado por la *API*, en referencia a los tipos de letras (*fonts*) que se disponen, ya que el uso de otro tipo requiere de una adaptación del software, la cual no forma parte del *API 2D*.

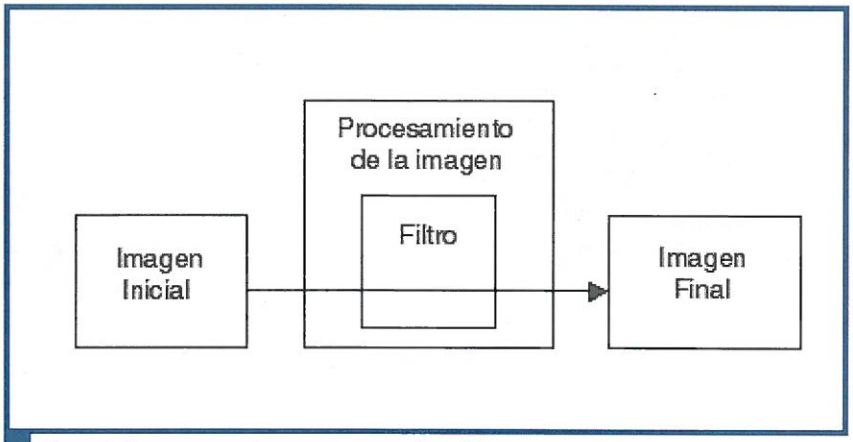


Figura 2. Procesamiento de una imagen con *Java 2D*.

FUNDAMENTOS PARA EL USO DEL API JAVA 2D

El modelo básico para el tratamiento de los gráficos no ha variado, ya que para gestionar un gráfico se genera un contexto gráfico y se invocan los métodos necesarios de la clase *Graphics* correspondientes. El uso de *Java 2D* al crear una clase *Graphics2D* derivada de *Graphics* permite que se puedan utilizar más atributos y un nuevo conjunto de métodos para tratar los gráficos, como se describirá más adelante.

La *API* automáticamente ajusta las diferencias entre los diferentes dispositivos, por lo que es igual que el destino sea un monitor o una impresora. A continuación se van a describir una serie de características elementales que permitirán utilizar la *API Java 2D*, a partir del cual podremos dibujar gráficos, visualizar imágenes, modificarlas, etc.

COORDENADAS DEL SISTEMA

Se utilizan dos tipos de sistemas de coordenadas:

- **User Space.**

El denominado *User space o* sistema de coordenadas que presentar, es un sistema de coordenadas lógicas, sobre un dispositivo independiente. Se podría decir que es el sistema de coordenadas de lo que queremos visualizar. Es utilizado exclusivamente por las aplicaciones y todo lo que se vaya a presentar al usuario es adaptado por las rutinas de la *API 2D* a este sistema.

Como se muestra en la figura 1, el origen de coordenadas del *user space* está localizado en la esquina superior izquierda del espacio definido, de tal forma que los valores <*x*> incrementan hacia la derecha y los valores de <*y*> hacia abajo.

Se puede decir que el *user space* representa una abstracción de todos los posibles sistemas de coordenadas con independencia del dispositivo que se va a utilizar en contra del *device space* que representa el sistema de coordenadas para un dispositivo en particular. A menudo, los *drivers* del dispositivo son los encargados de realizar esta transformación.

- **Device Space.**

Device space o sistema de coordenadas del dispositivo, es un sistema de coordenadas dependiente del dispositivo, es decir, en función del dispositivo que se va a utilizar, como por ejemplo puede ser una impresora. *Java 2D* define tres niveles de información sobre la configuración que son necesarios para las conversiones del *user space* al *device*

space. Esta información está gestionada por las siguientes clases:

- *GraphicsEnvironment*, como entorno gráfico.
- *GraphicsDevice*, como dispositivo gráfico.
- *GraphicsConfiguration*, configuración gráfica.

La clase *GraphicsEnvironment* contiene el conjunto de dispositivos que va a utilizar una aplicación *Java* en una determinada plataforma. Estos dispositivos suelen ser normalmente pantallas, impresoras y *buffers* de imágenes. Además esta clase incluye una lista con todos los tipos de letras disponibles.

Con esta nueva API se pueden dibujar superficies, curvas, arcos, etc.

GraphicsDevice describe el dispositivo visible a la aplicación, como una impresora o una pantalla. Cada posible configuración del dispositivo está representada por una instancia de *GraphicsConfiguration*. Por ejemplo, en un *SVGA*, se pueden utilizar varios modos gráficos: **800x600x256** colores, **1024x768x256** colores, etc. Entonces, la pantalla de esta *SVGA* se representa por un objeto *GraphicsDevice* y cada uno de sus modos por un objeto *GraphicsConfiguration*. Cada *GraphicsEnvironment* puede incluir uno o más *GraphicsDevice*, y cada uno de éstos puede contener uno o más *GraphicsConfiguration*.

Con todo esto, se puede tener la información necesaria para localizar y utilizar un dispositivo y convertir las coordenadas del *user space* al *device space*. Se debe destacar que *Java 2D* realizará todas las conversiones necesarias entre el *user space*, que a nivel de aplicaciones es el sistema de coordenadas que se utiliza para mostrar algo y el *device space*, que es donde verdaderamente se presentará. Esto se debe a que, como se mencionó anteriormente, el sistema de coordenadas de un monitor es diferente al de una impresora, un *plotter* o similar, por lo que gracias a *Java 2D* el proceso resulta de un modotan transparente.

TRANSFORMACIONES

La *API Java 2D* define un modelo de transformación de coordenadas en el que todas las transformaciones, incluyendo las referentes al *user space* y al *device space* son representadas por objetos del tipo *AffineTransform*, que determinan las condiciones de cambio de coordenadas utilizando una serie de matrices. Mediante una transformación de este tipo aplicada a un contexto gráfico se pueden realizar rotaciones, reescalados, desplazamientos o cortes en figuras, textos o imágenes cuando éstos son visualizados. Las transformaciones pueden ser aplicadas a cualquier objeto gráfico, teniendo en cuenta que se realizan cuando las coordenadas de *user space* son convertidas al sistema de coordenadas del *device space* mediante la invocación a un sencillo método.

IMÁGENES

Una imagen está compuesta por un conjunto de *pixels* organizados de una determinada forma. Un *pixel* o punto define la apariencia de una imagen en un determinado lugar. Por otra parte, una tabla de *pixels* es denominada *raster*. La apariencia de un punto se puede definir directamente por un conjunto de valores o mediante un índice a una tabla de colores de la imagen.

En las imágenes que contienen más de 256 colores, los *pixels* son representados por un color, un valor *alpha* (o nivel de transparencia), y otras características de visualización para cada posición de la pantalla. Las imágenes que se definen así suelen ocupar más espacio que las definidas mediante el sistema de color-indexado, pero tienen mucha mejor calidad y parecen más reales.

Mediante la clase BufferedImage se puede trabajar con una imagen

Con una imagen tratada con el sistema de color-indexado, cada color de la imagen está limitado a los colores especificados en una tabla de colores, que suelen ser los colores utilizados en la imagen. Con este sistema, se requiere menos espacio para la imagen y suelen ser utilizados típicamente para las imágenes de 16 ó 256 colores. En *Java 2D* las imágenes se componen de dos partes diferentes.

- La imagen en *pixels*.
- La información para tratar los *pixels*.

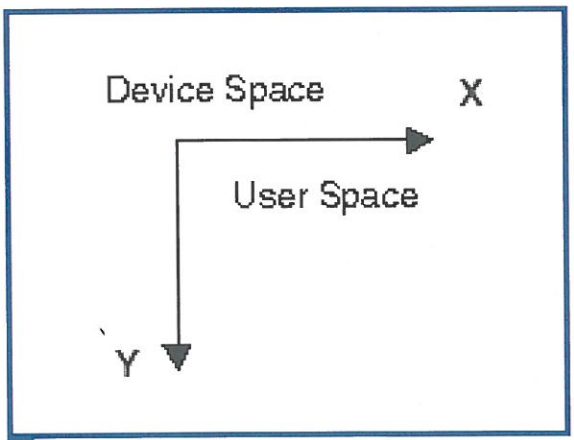


Figura 3. Sistema de coordenadas inicial.

UTILIZACIÓN DEL API DE JAVA 2D

Una vez que se ha definido el modelo utilizado por *Java 2D API*, así como sus principales características, clases, interfaces y operaciones disponibles, vamos a desarrollar una serie de ejemplos elementales para empezar a familiarizarnos con el uso de la *API Java 2D*.

Mediante AffineTransform se realizan las transformaciones necesarias entre los sistemas de coordenadas, rotaciones, etc

En el siguiente ejemplo se utiliza un objeto de tipo *Graphics2D* para dibujar unos rectángulos en pantalla. Como se va apoder observar, no se utiliza una instancia de ningún objeto de este tipo, sino que se utiliza la instancia de tipo *Graphics* que es pasada al método *paint()*.

```
import java.awt.*;
public class e1 extends Frame {
  public e1 () {
      super ("Ejemplo Java 2D");
      setSize (300,300);
      setVisible (true );
  }
  public void paint ( Graphics g ) {
      Graphics2D mi2d = (Graphics2D) g;
      for ( int x = 100 ; x > 50 ; x -= 10 ) {
        mi2d.setColor ( Color.blue );
        mi2d.drawRect ( x, x, 10+x, 10+x);
      }
  }
  public static void main ( String args[] ){
      new e1();
  }
}
```

Hasta ahora no se ha hecho uso de las verdaderas capacidades *2D* de *Java*. En el siguiente ejemplo, que mostramos, vamos a ver cómo añadir algunas de sus tan novedosas características, como por ejemplo son la transformación y el uso de *GeneralPath,* que nos va a permitir el poder especificar una figura que hemos elegido segmento a segmento.

```
public void paint ( Graphics g ) {
  Graphics2D mig2d = (Graphics2D) g;
  GeneralPath misegm = new GeneralPath (
      GeneralPath.WIND_EVEN_ODD);
  misegm.moveTo ( 0.0f,  0.0f);
  misegm.lineTo ( 0.0f,  75.0f);
  misegm.lineTo ( 100.0f, 75.0f);
  misegm.lineTo ( 100.0f, 0.0f);
  misegm.closePath();

  AffineTransform at = new
AffineTransform ();
  at.setToRotation ( -Math.PI /
6.0 );
  mig2d.transform (at);
  at.setToTranslation ( 50.0f,
200.0f );
  mig2d.transform (at);
  mig2d.setColor ( Color.green
);
  mig2d.fill ( misegm );
}
```

En este ejemplo aparecerá un rectángulo al que se le ha aplicado una rotación. Mediante el uso de *Java 2D* podemos definir cualquier tipo de figura usando *java.awt.Shape* y de ahí la clase *GeneralPath()*, gracias a la cual hemos dibujado el rectángulo. Pero lo más interesante del ejemplo reside en el uso de *java.awt.geom.AffineTransform* que nos permite realizar la rotación y cambio de origen.

Java 2D utiliza dos tipos de sistemas de coordenadas, los denominados User Space y Device Space

Recordando todos los conceptos anteriormente descritos, *AffineTransform* se encarga de realizar todo el trabajo para adaptar el *device space* a nuestro *user space* y poder de esta manera colocar nuestra imagen donde deseemos y a la vez ejecutar la rotación. Veamos este proceso con algo más de detalle.

Inicialmente, el *device space* y el *user space* son coincidentes, es decir, el sistema de coordenadas coincide. Tendríamos un sistema de coordenadas como indica la figura 3.

Una vez ejecutadas las sentencias citadas a continuación:

```
at.setToRotation ( -Math.PI / 6.0 );
mig2d.transform (at);
```

Se realiza una rotación al *user space*, por lo que gráficamente lo podemos observar en la situación de la figura 4.

Esta rotación se debe expresar en radianes, por lo que **(-PI/6.0)** es equivalente a **–30** grados o 30 grados en sentido contrario a las agujas del reloj. A continuación se aplicará un cambio de origen de coordenadas, por lo que modificaremos el origen del

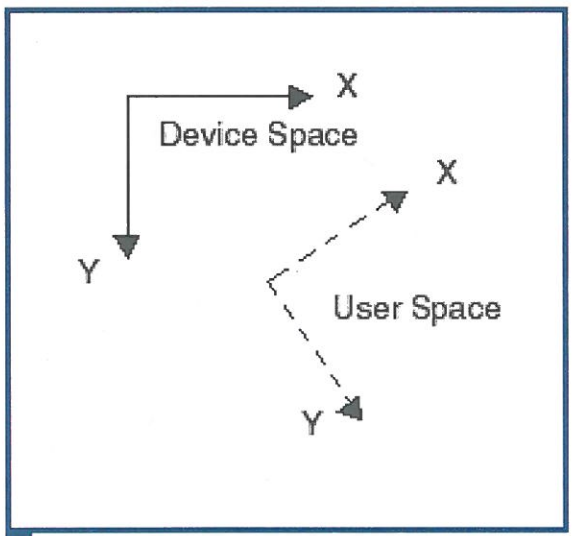


Figura 5. Cambio de origen del User Space.

user space, a las coordenadas que definíamos respecto del *device space*, en nuestro caso tendremos:

```
at.setToTranslation ( 50.0f, 200.0f );
mig2d.transform (at);
```

En la figura 5 de este artículo se muestra el estado final de los sistemas de coordenadas anteriormente a visualizar la imagen.

Una imagen se compone de una serie de pixels y de la información necesaria para gestionarlos

Para finalizar con el proceso nuestra figura será asignada a un color, también se rellenará, visualizándose de la siguiente manera:

```
mig2d.setColor ( Color.green );
mig2d.fill ( misegm );
```

Como hemos podido observar en este ejemplo mostrado, si no perdemos de vista los campos en los sistemas de coordenadas, la representación gráfica nos puede resultar de un modo relativamente sencillo y práctico de llevar a cabo.

CONCLUSIONES

A lo largo de este artículo se han mostrado los conceptos básicos para el manejo de la nueva *API Java 2D*, que permite realizar todo un conjunto de nuevas operaciones sobre gráficos, imágenes, texto, etc.

Todo esto ha sido con el objeto de ofrecer un entorno de desarrollo multimedia de utilización muy sencilla, práctica y eficaz.

En el siguiente artículo comprobaremos como visualizar imágenes, rotarlas y realizar las transformaciones básicas con filtros.

BIBLIOGRAFÍA

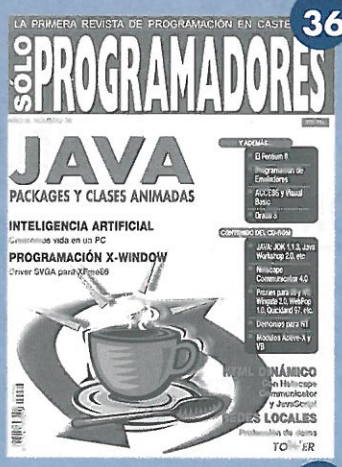
- "Java Second Edition". Michael Morrison. QUE.
- Programmer's Guide to the *Java 2D API*. Sun Microsystems.

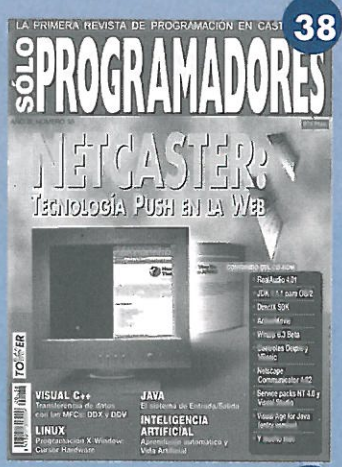
El Tercer Mundo está desapareciendo.
Enhorabuena.
Teléfono: 902 402 404 - www.ayudaenacción.com
Porque su futuro ya está cambiando. Porque están aprovechando sus propios recursos. Porque, trabajando juntos, estamos contribuyendo para que dejen de ser el Tercer Mundo.
Desde 1981
Ayuda en Acción
Sembrando Futuro

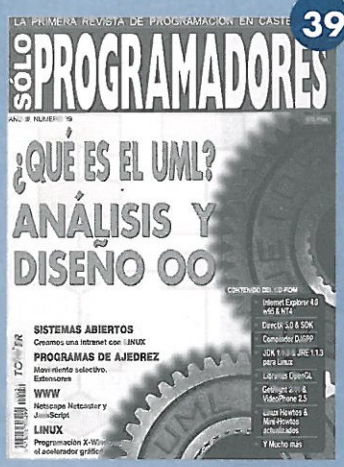
números atrasados

SÓLO PROGRAMADORES

completa ya tu colección







suscríbete

a Sólo Programadores

y consigue un **magnífico descuento**

suscripción **normal**

ahorro **20%**

12 revistas (1 año) por sólo...

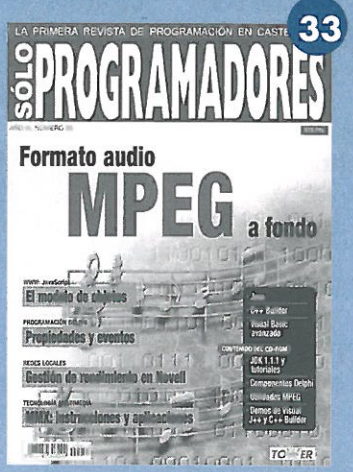
9.350 ptas.

suscripción **estudiantes** (carreras técnicas)

ahorro **40%**

12 revistas (1 año) por sólo...

7.050 ptas.



Estudio práctico del TCP

Juan Manuel Menéndez (jfrias@ibm.net)

Protocolo: Regla de urbanidad o cortesía establecida por decreto o costumbre. (Según el Diccionario de la Lengua de la Real Academia Española). Protocolo: Conjunto de reglas que rigen el comportamiento de las conexiones para transferencias de los datos entre dos dispositivos. (Según un diccionario de comunicaciones).

ANÁLISIS DEL PROTOCOLO TCP

Detrás de una conexión a *Internet*, del acceso a un disco de red o de una transferencia *ftp* se ocultan una serie de reglas que permiten que esta información viaje correctamente entre ambos extremos de la conexión. Es objetivo de este artículo introducir de forma práctica los principios básicos del protocolo *TCP*. La plataforma sobre la que se ha realizado el artículo consiste en:

- Ordenador con *Windows 95*.
- Ordenador con *Linux*.
- Una conexión *Ethernet*.
- Dos programas *C* en el servidor *Linux*.
- Un programa en *Visual Basic* en el cliente *Windows*.
- El programa **TCPDUMP** que permite el análisis de los segmentos *TCP*.

Para aquellos lectores que no puedan disponer de una configuración similar a la utilizada para el desarrollo de las pruebas he intentado conseguir que las capturas de las pantallas cubran en su totalidad las partes del protocolo que se explican en los distintos apartados del capítulo. Pero, de todos modos si se dispone de una plataforma con *Linux se* podría utilizar *tcpdump* y procesar el fichero llamado **tcpdump.out** que se incluye en el *CD-ROM* de la revista. Más adelante se procederá a explicar cómo se puede realizar la lectura de este fichero de una manera bastante sencilla.

INSTALACIÓN DEL PROGRAMA TCPDUMP

El programa **tcpdump** está disponible para distintas versiones del sistema operativo de tipo *Unix* como por ejemplo pueden ser *Solaris*, *Aix*, *SCO*, *Linux*, etc. Y puede ser descargado desde *Internet* de la dirección **ftp://ftp.ee.lbl/gov.tcpdump.tar.z**, lo encontrará con un formato *tar* comprimido.

Tabla 1. Representación de los flag en la salida de tcpdump.

S	SYN	Sincroniza los números de secuencia.
F	FIN	El emisor ha finalizado el envío de datos.
R	RST	Restaura la conexión.
P	PSH	Los datos deben de ser enviados al otro extremo.
.		Sin flag.

El proceso de instalación que se va a describir a continuación es para el sistema *Linux*, siendo los requisitos necesarios para instalar el programa los siguientes:

- Version del *kernel* superior a **2.0.25**.
- Compilador *gcc*.
- Biblioteca de funciones *libcap*.

La biblioteca de funciones *libcap* puede ser descargada de la dirección: **ftp:// ftp.ee.lbl.gov/libpcap.tar.Z**

Este fichero también viene en formato *tar* comprimido y se pone en formato estándar con los comandos:

```
uncompress libcap.tar.Z
```

El programa tcpdump está disponible para distintas versiones de sistemas operativos de tipo Unix

El resultado de este comando es el fichero **libcap.tar,** que está en formato tar. Para obtener los ficheros incluidos en el mismo de se debe ejecutar el comando:

```
tar -xvf libpcap.tar
```

Este comando vuelca sobre el directorio **pwd/libcap-0.4** (*pwd* es el directorio sobre el que se está situado en el momento de lanzar el comando *tar*), unos ficheros con los que se construirá la biblioteca de funciones. A continuación debemos editar el fichero **Makefile.in** y configurar los directorios para los ficheros *include*, páginas de manual y la propia biblioteca de funciones. Una vez configurados los parámetros, tendremos que ejecutar elsiguiente comando:

```
./configure
```

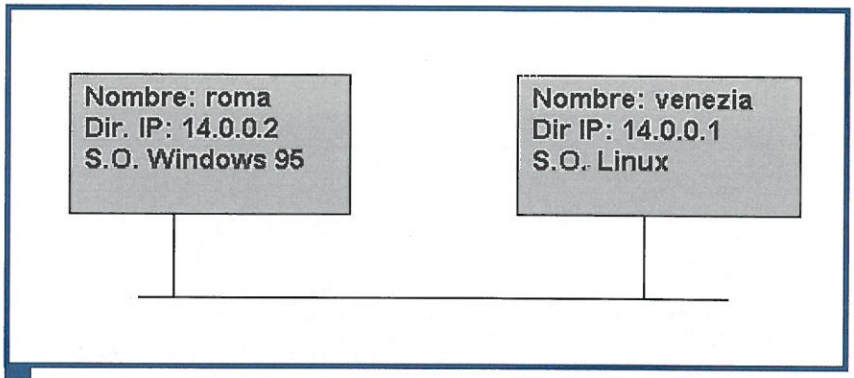


Figura 1. Esquema de configuración de las prácticas.

```
make install
make install-incl
make install-man
```

Si sólo se desea instalar *libpcap* para trabajar con *tcpdump* lo más fácil es hacer que el directorio de ambos compartan el mismo directorio raíz y dejar que se instale automáticamente en el proceso de instalación de *tcpdump*. Lo primero que hay que hacer para instalar *tcpdump* es descomprimir el fichero **tcpdump_tar:Z**:

```
uncompress tcpdump_tar.Z
tar -xvf tcpdump_tar
```

Una vez que ambos ficheros han sido descomprimidos se edita el fichero **Makefile.in** del directorio de tcpdump y se da valor a los parámetros:

- BINDEST = nombre del directorio donde se va a colocar el programa.
- *MANDEST* = nombre del directorio para las páginas del manual. (*usr/man* normalmente)

Una vez que han sido asignados los parámetros requeridos debemos ejecutar:

```
./configure
```

Este *script nos* confeccionará un fichero llamado *Makefile* el cual intentaremos procesar mediante los siguientes comandos:

```
make install  (para el programa)
make instal-man  (para las páginas de manual)
```

Si todo ha ido correctamente tendremos nuestro programa preparado para ejecutarse, aunque de todas formas antes de proceder a ejecutar los programas de configuración se puede encontrar información en los ficheros **README** e **INSTALL** de los directorios. Aunque la instalación de las páginas del manual es opcional, contienen tanta información sobre este programa que casi cabría considerarlas de instalación obligatoria.

PROGRAMAS DE COMPLEMENTO

Además del programa **tcpdump**, para poder seguir bien el artículo se han creado tres programas que permiten modificar el tipo de transmisión de datos. Estos programas son:

- **tcptest.** Programa escrito en *C* que queda arrancado como un demonio, crea un *socket* y queda a la escucha de las peticiones de conexión lanzadas desde programas clientes. Este programa es un servidor multiproceso y puede atender varias sesiones simultáneamente. Acepta tres parámetros:
  - Número de puerto.
  - Longitud del paquete a recibir.

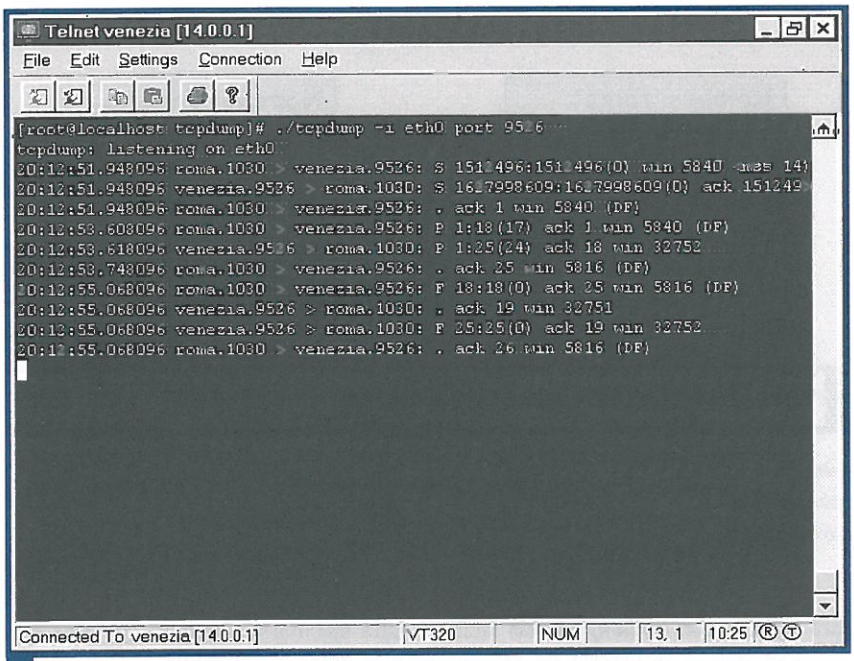


Figura 2. Salida del programa TCPDUMP.

- Intervalo transcurrido entre la recepción de los datos y la respuesta.

- **tcplci**: Programa escrito en *Visual Basic*. Dispone de cuatro botones:
  - **Inicio**: Inicia la conexión con el *socket*.
  - **Cerrar**: Cierra la conexión con el *socket*.
  - **Envio**: Manda un paquete de **17** octetos al servidor.
  - **Envio Largo**: Manda un paquete cuyo valor es igual al número de **K-octetos** seleccionados en la lista de la derecha.
- **tcpparm**: Programa que modifica la longitud del paquete que va a recibir y el intervalo de espera de forma dinámica, sin necesidad de parar y arrancar el programa. Estos valores pueden ser modificados dinámicamente incluso con la sesión ya establecida. El utiliza para comunicarse con el programa **tcptest** la memoria compartida.

Los programas **tcptest** y **tcparm** se deberán instalar en el servidor *Unix* (han sido probados en sistemas *Linux* y *AIX*), y para instalarlos lo único que hay que hacer es editar el fichero *Makefile* y dar valores a los directorios según se indica en el propio fichero.

El *Makefile* a partir de directorio indicados en *WORKDIR* dispone de los siguientes directorios:

- *$WORKDIR/bin* para el ejecutable.
- *$WORKDIR/etc* para los ficheros gestores de *IPC*.
- *$WORKDIR/log* para el fichero de *log*.

Para instalar esto es necesario ejecutar:

```
make todo
```

Otras opciones vienen detalladas en el propio fichero.

El programa **tcpcli** está en *Visual Basic 5* (Para *Windows 95/98*, no está probado sobre *NT*), y se suministra tanto en formato instalable mediante **setup**, como con los fuentes para ser compilados.

ARRANQUE DE LOS PROGRAMAS

El programa **tcptest** se arranca mediante la secuencia de comandos:

```
export WORKDIR= directorio WORKDIR de la
      instalacion
tcptest  -d num_puerto [ -t num_octetos ] [ -i
      intevalo].
```

El único parámetro obligatorio es el número de puerto, ya que los otros son opciones y asume los valores **256** y **0** para -y e -i respectivamente. Si no se carga la variable de entorno *WORKDIR* el programa asume que se trata del directorio */tmp* y busca en */tmp/etc* los ficheros necesarios, cancelando por *abend* si no los encuentra. Otro tanto ocurre si no se codifica correctamente dicha variable.

El programa arranca como demonio, por lo que deja libre el teclado, aunque lanza mensajes a través de la pantalla. Para pararlo se utiliza el comando *kill*.

TCP es usado por otros protocolos de aplicación como son telnet, FTP ó HTTP

Los parámetros se pueden modificar en cualquier momento mediante el comando:

```
tcptest [ -t num_octetos ] [ -i intevalo].
```

También en este caso es necesario tener la variable *WORKDIR* con el valor correcto.

El programa **tcpcli** se procede a arrancar de la misma manera que se haría con cualquier otro programa *Windows*.

ARRANQUE DEL PROGRAMA TCPDUMP

El programa **tcpdump** se arranca desde una *shell* mediante el comando:

```
tcpdump [opciones.....]
```

Las opciones de arranque se suministran como parámetros y permiten configurar qué tipo de paquetes *TCP* ó *UDP* se van a monitorizar. También permite discriminar el tipo de protocolo, número de puerto, formato de la salida, etc. Para obtener una descripción completa de las opciones de arranque es posible recurrir a las páginas del manual. Para seguir el artículo se arrancará con:

- *i* nombre de la interfaz. Puede ser *eth0*.
  port número de puerto que debe de coincidir con el valor suministrado en el parámetro *-d* del programa **tcptest.**

Cuando el receptor no puede seguir recibiendo datos pone el tamaño de su ventana a cero

La sentencia puede quedar:

```
tcpdump -i eth0  port 7007.
```

Aquellos lectores que deseen procesar el fichero **tcpdumd.out** deberán ejecutar el siguiente comando:

```
tcpdump -r ./tcpdump.out
```

En este caso no es necesario especificar otros parámetros pues estos han sido previamente especificados al grabar el fichero con el comando:

```
tcpdump -i eth0   -r ./tcpdump.out  port 7007.
```

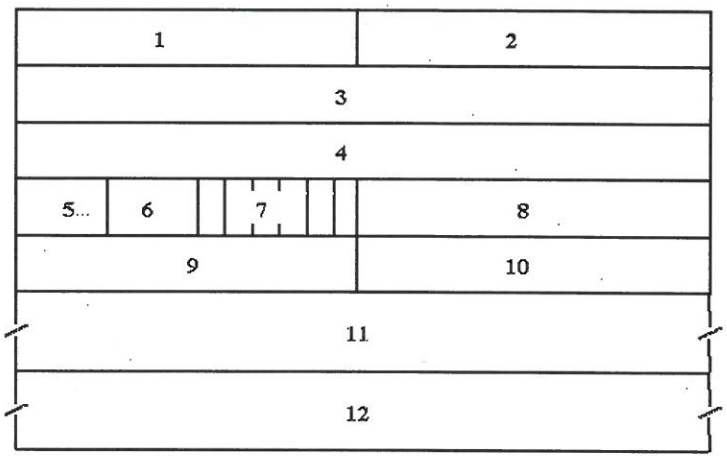
SALIDA DEL PROGRAMA TCPDUMP

La salida sobre pantalla del programa tiene un aspecto similar al de la figura 2, en donde cada línea representa un paquete *TCP/UDP*. El primer campo indica la hora, el segundo la dirección *ip* y el puerto origen de los datos. A continuación viene un carácter que representa gráficamente cuál es el sentido de los datos, seguido por la dirección ip y el puerto del destino del segmento. Estos campos se obtienen de la cabecera *ip* más los primeros 32 octetos de la cabecera *TCP*.

A continuación aparecen los *flags*, que pueden tener los valores que se describen en la tabla 1. Los números de secuencia siguen a continuación de la forma *primer_numero:ultimo_numero(longitud)*.

En la primera trama y en las siguientes tramas los contadores de primer y ultimo número aparecen con el valor relativo al origen, debido a que el programa **tcpdump** formatea la salida de esta forma, pero del mismo modo sería posible hacer que genere siempre los números de secuencia reales.

A continuación de los números de secuencia sigue otra información que se verá más adelante.



1) Número de puerto de origen. Longitud 16 bits.
2) Número de puerto de destino. Longitud 16 bits.
3) Número de secuencia. Longitud 32 bits.
4) Número de secuencia al que se acusa recibo. Longitud 32 bits.
5) Número de palabras de 4 octetos dela cabecera. Longitud 4 bits. Este campo incluye la longitud de los campos 11 y 12 que son de longitud variable. En el caso mínimo son 20 octetos (5 palabras) con los campos 11 y 12 vacíos.
6) Reservados.
7) Flags. 6 campos de 1 bit cada uno y son:
   - URG: Segmento urgente. Salta otros segmentos en cola.
   - ACK: Se han acusado recibo al segmento cuyo número de secuencia está indicado en el campo 4.
   - PSH: Los datos deben ser pasados al receptor lo antes posible.
   - RST: Restaura la conexión.
   - SYN: Sincroniza los números de secuencia para iniciar una conexión.
   - FIN: El emisor ha terminado el envío de datos.
8) Tamaño de la ventana. Longitud 16 bits.
9) Checksum. Longitud 16 bits. Resultado de la operación de control de paridad que se hace sobre la cabecera y datos del segmento. Esta operación la realiza el emisor y la verifica el receptor.
10) Puntero de segmento urgente. Longitud 16 bits. Este campo se usa cuando el flag URG está activado, este método se utiliza para hacer que los segmentos urgentes salten todas las colas que se puedan formar tanto en el origen como en el destino.
11) Opciones de transmisión.
12) Datos.

Figura 3. Esquema de la cabecera TCP.

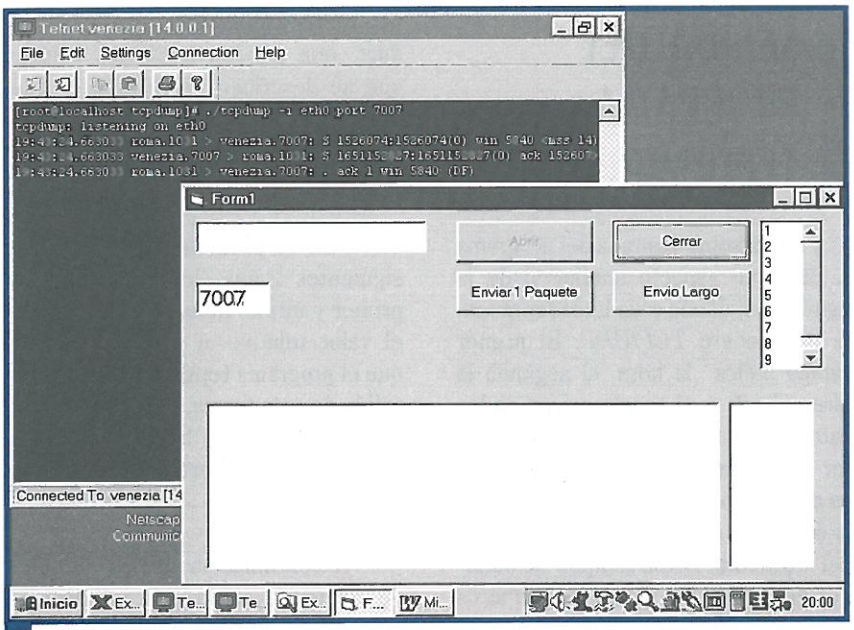


Figura 4. Aspecto del escritorio Windows tras el establecimiento de conexión.

TCP: TRANSMISION CONTROL PROTOCOL

El *TCP* es un protocolo orientado a conexión que permite el intercambio fiable de información entre dos extremos de aplicación utilizando para ello sistemas de transmisión de datos. Por ser un protocolo orientado a transmisión lo primero que debe establecerse entre ambos extremos de la sesión es una conexión. Para establecer una sesión *TCP* se necesitan dos programas, ya que *TCP* no soporta sesiones de *broadcast*.

El intercambio fiable de información quiere decir que todos los segmentos de información que envía un extremo son recibidos en el orden adecuado en el otro. Para conseguir esta fiabilidad el tráfico *TCP* se resume en lo siguiente:

- Los datos de la aplicación son divididos en segmentos.
- Cuando se envía un segmento se mantiene un temporizador, esperando que el otro extremo acuse recibo del segmento enviado. Si expira el plazo del temporizador sin recibir la confirmación al segmento, éste es retransmitido.
- Cuando el segmento llega al otro extremo de la conexión, el *TCP* envía el acuse de recibo.
- Se mantiene una verificación por *checksum* para verificar que los datos no han sido modificados durante la transmisión. Si el proceso de *checksum* falla se descarta la trama y no se manda acuse de recibo. Como consecuencia de esto al expirar el intervalo en el emisor, el segmento vuelve a retransmitirse.
- Los números de secuencia que lleva cada segmento son usados por el receptor para entregar los datos a la aplicación en el orden adecuado.
- Como ambos extremos de la conexión pueden tener distintas características, el *TCP* tiene unos mecanismos de control de flujo que permiten controlar que el extremo no inunde al otro con más información de la que es capaz de tratar.

Otra característica es que el protocolo *TCP* envía bloques de octetos sin separación entre ellos. Por ejemplo un programa puede escribir sobre un *socket* **10** octetos, después **30** octetos y a continuación **100** octetos. El otro extremo va a recibir **140** octetos pero no necesariamente en la forma en que los ha escrito el emisor, ni va a existir un marcador interno que separe cada bloque de escritura. Todo este tipo de gestión debe ser realizada por los protocolos de nivel superior.

Como se puede ver en la figura 3 cada segmento *TCP* contiene los números de los puertos origen y destino. Estos dos valores junto con las direcciones *IP* de destino y origen que van en la cabecera *IP* identifican biunívocamente cada conexión.

El número de secuencia identifica el primer octeto de información en el segmento. Cuando se establece una conexión se envía un segmento con el

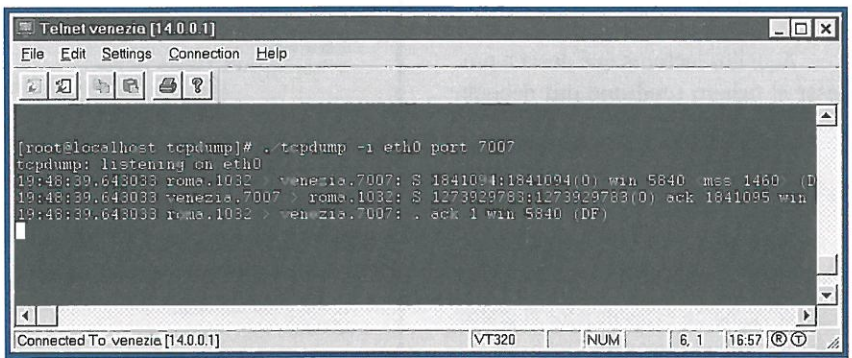

Telnet venezia [14.0.0.1]
File Edit Settings Connection Help
[root@localhost tcpdump]# . tcpdump -i eth0 port 7007
tcpdump: listening on eth0
19:48:39.643033 roma.1032 > venezia.7007: S 1841094:1841094(0) win 5840 mss 1460 (D
19:48:39.643033 venezia.7007 > roma.1032: S 1273929783:1273929783(0) ack 1841095 win
19:48:39.643033 roma.1032 > venezia.7007: . ack 1 win 5840 (DF)
Connected To venezia [14.0.0.1]
VT320
NUM
6, 1
16:57

Figura 5. Pantalla de TCPDUMP al inicio de la sesión.

flag SYN activado conteniendo el número inicial de secuencia. Cuando el receptor de un segmento acusa recibo envía un segmento indicando el número de secuencia que espera recibir en el siguiente paquete. Como se puede deducir cada octeto de la transmisión está numerado y este mecanismo de numeración es el que impide que los segmentos se puedan perder o ser entregados en distinto orden al que fueron escritos por la aplicación emisora.

La longitud de la cabecera *TCP* viene en un campo de 4 octetos. Esta longitud viene dada en palabras de **32** *bits* (4 octetos). Como con 4 octetos el máximo valor que se tiene es **15**, el máximo tamaño de una cabecera *IP* es de **60** octetos (4 * 15). El mínimo tamaño que tiene una cabecera *TCP* es de **20** octetos, de tal forma que el resto hasta los **40** son las opciones y tienen una longitud variable.

El emisor envía el flag Push cuando el buffer de salida está vacío

La ventana es utilizada para el control de flujo; se puede definir una ventana como el máximo número de octetos que un receptor está dispuesto a admitir. Cuando un extremo receptor de datos no puede procesar los datos a una velocidad suficiente, lo que hace es disminuir su ventana. En el protocolo *TCP* la ventana disminuye si el receptor no acusa recibo a los segmentos recibidos.

Mediante los mecanismos que se han descrito de forma general, el *TCP* suministra un servicio de transporte orientado a conexión y fiable. *TCP* divide la información en segmentos, activando un temporizador cada vez que envía un segmento. En el extremo receptor, los datos son recibidos, reordenados si llegan desordenados y se rechazan los segmentos repetidos y los alterados.

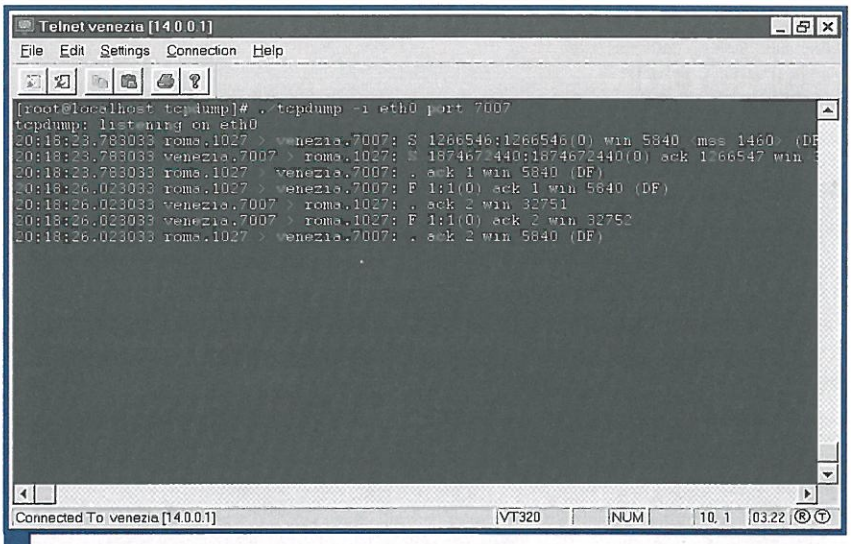


Figura 6. Secuencia entre inicio y cierre de conexión.

TCP es usado por otros protocolos de aplicación como pueden ser *telnet*, *FTP*, *HTTP*.

ESTABLECIMIENTO Y FINALIZACIÓN DE LA CONEXIÓN

En el protocolo *TCP* antes de empezar el envío de datos de un extremo a otro se debe de establecer una conexión entre los programas. Lo primero que vamos a hacer es poner el entorno de pruebas arrancando en el servidor el programa **tcptest** de la forma:

```
tcptest -d 7007 -t 256
```

En otra sesión de *telnet* haremos los mismo con el programa **tcpdump** de la forma:

```
tcpdump -i eth0 port 7007
```

El valor del parámetro *-i* será el del adaptador *ethernet*, que normalmente será **eth0**. En el cliente *Windows* se activará el programa **tcpcli**, y una vez realizados todos estos pasos el escritorio de *Windows* tendrá un aspecto similar al de la figura 4.

Iniciamos la conexión con el servidor pulsando el botón **Abrir** del formulario, y una vez establecida se puede observar en la sesión de telnet con la salida de **tcpdump**, cómo se ha establecido el inicio de la sesión.

- En el primer segmento el cliente envía un *SYN* especificando el número de puerto con el que desea conectar y el número de secuencia inicial.
- El servidor responde con otro segmento *SYN* que contiene el número de secuencia inicial, y le manda un *ACK* (acuse de recibo) indicando cuál es el siguiente número de secuencia que espera recibir.
- El cliente responde al segmento *SYN* del servidor con un *ACK*.

Estos tres segmentos completan el establecimiento de una conexión *TCP*, en la que el extremo que envía el primer *SYN* realiza una apertura activa de la conexión y el otro realiza la apertura pasiva. Se puede comprobar cómo queda establecida la sesión ejecutando en el servidor el comando:

```
netstat -an | grep 7007
```

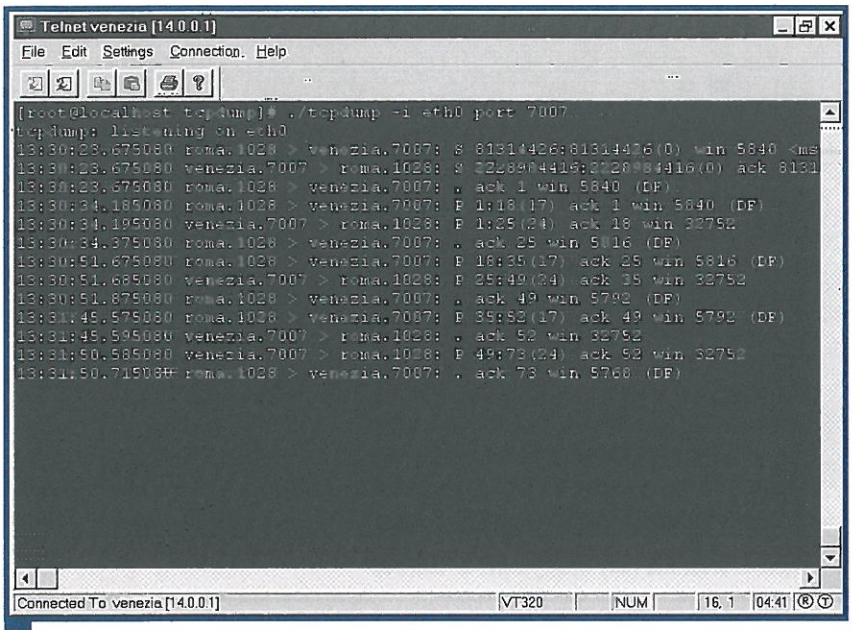


Figura 7. Esquema de la transmisión normal.

En el cliente se obtiene un resultado similar tecleando desde una ventana de *DOS*:

```
netstat -an
```

Para cerrar la sesión se pulsa el botón **Cerrar** del formulario del programa **tcpcli**, y vemos en la sesión de **tcpdump** el resultado, que puede observarse en la figura 5.

El inicio de una conexión está formado por dos segmentos SYN y un ACK

El cierre de la sesión está compuesto por cuatro segmentos, donde cada par se denomina medio-cierre (*half-close*). Como el protocolo *TCP* es una conexión *full-duplex*, es decir, que la información puede viajar en cualquiera de los dos sentidos, cada uno de estos sentidos debe cerrar su media sesión de forma independiente. Cada extremo envía un segmento *FIN*, y este segmento es originado cuando la aplicación hace un cierre sobre el *socket*. En el momento en que el otro extremo de la aplicación recibe el segmento *FIN* devuelve un *ACK* y sabe que ya no va a recibir más segmentos, pero puede seguir mandando información, es decir se ha completado un medio cierre. El otro medio cierre se realiza de igual forma, de tal forma que se completa el cierre de la conexión. Una vez realizado el segundo medio-cierre ya aparece como resultado del comando *nestsat*. En la figura 6 se puede observar las secuencias de inicio y cierre de una conexión.

TRÁFICO DE DATOS

Una vez visto el protocolo de establecimiento y cierre de una conexión TCP, vamos a ver qué ocurre cuando se transmiten datos a través de la misma. Hay dos clases de tráfico de datos: por un lado existe un tráfico caracterizado por el envío de unidades lógicas de información cada una de las cuales está incluida íntegramente en un segmento. Además un segmento sólo contiene una unidad de información y con una relación de **envío/tiempo no envío** baja. A este tipo de tráfico se le denomina interactivo y es propio de las sesiones *telnet*, *rexec*, *rlogin*. En el otro extremo se encuentra el tráfico caracterizado por todo lo contrario: unidades de información dividida en varios segmentos, y una relación **tiempo de envío/tiempo no envío** alta, es decir alto consumo del ancho de banda del medio de transmisión.

Estudios realizados sobre el tráfico *TCP* dan cuenta de que aproximadamente un 50% de los segmentos *TCP* son interactivos y el otro 50% es masivo. Pero si se cuenta el número de octetos transmitidos las estadísticas dicen que el 90% de los octetos que viajan por una red son de tráfico masivo y el resto, es decir el 10%, de tráfico interactivo. Lo anteriormente mencionado da una idea de lo diferentes que son ambos tipos de tráfico.

A continuación se verá como *TCP* maneja ambas situaciones. Para poder ver de una forma práctica lo que se va a explicar a continuación, vamos a establecer nuevamente nuestro punto de observación del trafico *TCP* como se ha hecho anteriormente y vamos a iniciar una nueva conexión.

TRÁFICO INTERACTIVO

Una vez establecida la conexión se envía un paquete pulsando el botón **Enviar 1 paquete** y observamos en la sesión de **tcpdump** qué ha ocurrido. Después del conocido protocolo de establecimiento de la sesión se puede ver como se manda un segmento de datos de longitud **17**, y al mismo tiempo el cliente acusa recibo con un segmento *ACK 1*. El servidor procesa los datos recibidos y devuelve una respuesta de longitud **24** y en el mismo segmento acusa recibo del segmento de datos recibido con *ACK 18*. Al reci-

bir esta información, el cliente acusa recibo de estos datos mandando un segmento sin datos únicamente con *ACK 25*. Si repetimos el envío se ve como se repite la secuencia:

- Un segmento de datos.
- Un segmento con la respuestas.
- Un segmento acusando recibo a la respuesta.

Si observamos los tiempos que aparecen a la izquierda se observa que cada secuencia ocurre en un intervalo de dos décimas de segundo. El proceso que realiza el servidor consiste en calcular un número aleatorio y devolverlo en la respuesta, para diferenciar las repuestas.

El TCP es un protocolo orientado a transmisión

Pero supongamos que el proceso que debe realizar es más pesado, por ejemplo una consulta a una base de datos. Además en una red real existen otros factores que hacen aumentar el tiempo de respuesta como es la existencia de varios equipos utilizándola simultáneamente. Para simular un entorno de estas características vamos a retrasar el envío de la respuesta a cinco segundos mediante el comando:

```
tcpparm -i 5
```

Repetimos el envío del paquete y observamos que se han enviado **4** segmentos y que entre los dos primeros segmentos se ha intercalado un segmento *ACK 52* acusando recibo al primer segmento de datos. La explicación a esta diferencia es que el *TCP* ha estado esperando la respuesta por parte del programa servidor para intentar aprovechar en el mismo segmento el acuse de recibo y la respuesta a la aplicación con el objetivo de intentar consumir la menor cantidad posible de ocupación del ancho de banda. No olvidemos que a pesar del avance en los sistemas de comunicaciones, las líneas siguen siendo el cuello de botella en los sistemas informáticos. Pasado un cierto intervalo y como la aplicación no suministra la respuesta, envía un segmento *ACK* para acusar recibo al paquete. A este tipo de acuse de recibo se le denomina "acuse de recibo retardado". Un segmento no puede quedar a la espera de acuse de recibo indefinidamente, pues cuando se envía un segmento el emisor activa unos temporizadores y pasado un intervalo de tiempo efectúa la retransmisión del segmento si no ha recibido acuse de recibo al segmento enviado.

TRÁFICO MASIVO

El tráfico masivo según se ha dicho anteriormente es aquél tipo de tráfico que tiende a consumir ancho de banda y que utilizan los protocolos de aplicación como *FTP*. Para estudiar cómo funciona este protocolo vamos a iniciar una nueva conexión y dentro de los tres segmentos que se intercambian en el inicio de la conexión se puede observar que en los segmentos *SYN* ambos extremos de la conexión intercambian el máximo tamaño de segmento que desea recibir. Este valor conocido por sus siglas inglesas *MSS* (*Maximum Segment Size*), es enviado por cada extremo para este parámetro y está contenido dentro de la parte de opciones de la cabecera *TCP*.

Vamos a realizar un envío de un bloque de datos de **2K** mediante el botón **Envío Largo**, habiendo seleccionado previamente el valor **2** de la caja de lista del formulario. Volvemos a observar lo ocurrido en la sesión de *telnet* donde se encuentra en arrancado **tcpdump**.

En primer lugar se debe apreciar que el tamaño de bloque de información que el cliente envía al servidor es mayor que el valor de *MSS*, por lo que *TCP* va a realizar una división de este bloque de información en tantos segmentos como sea necesario. En el caso de *2K* (**2048**) octetos el primer segmento de datos tendrá un tamaño de **1460** octetos y el segundo de **588** octetos (2048 - 1460). Después del envío del primer segmento el TCP del equipo servidor acusa recibo de este bloque de información mediante el "acuse de recibo retardado", el cliente manda los datos que faltan en otro segmento y el servidor manda el acuse de recibo al segmento. Para recibir finalmente una respuesta de **144** octetos.

Para poder entender mejor esta última parte de la transmisión vamos a

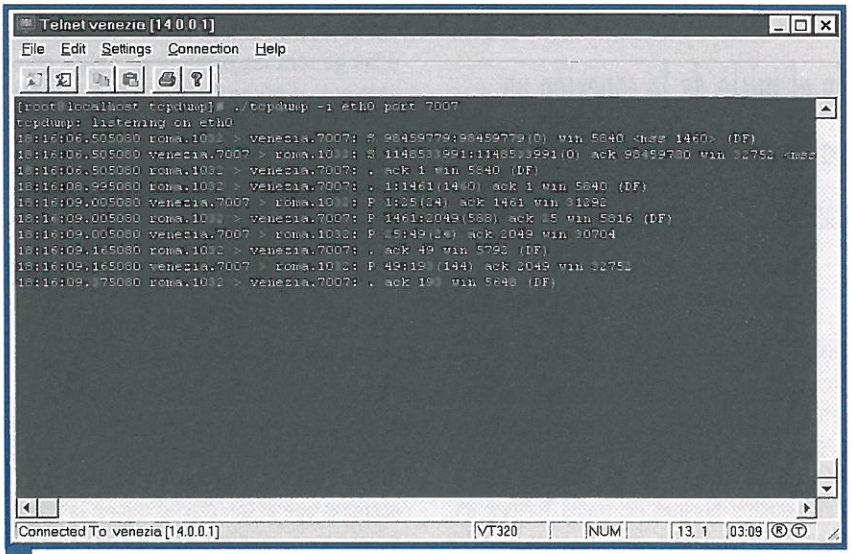


Figura 8. Esquema de la transmisión con ACK retrasado.

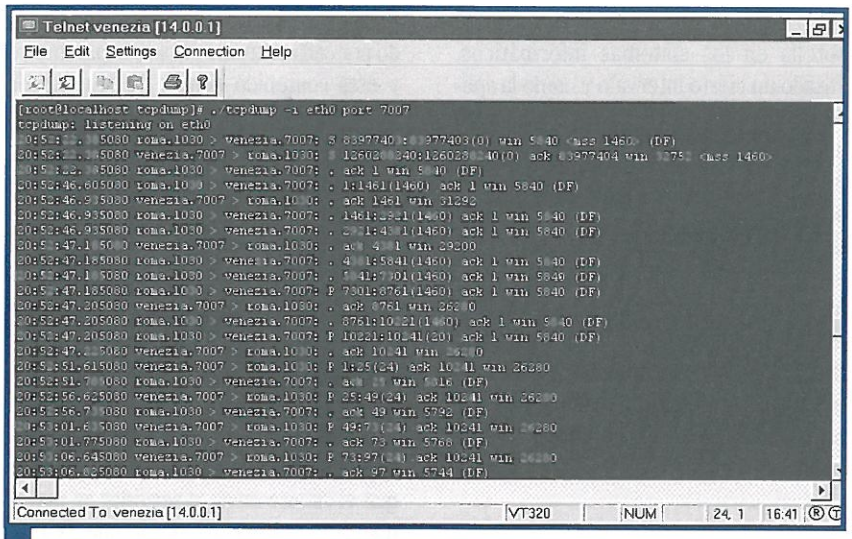


Figura 9. Esquema de la transmisión observando el cierre de una ventana.

ver cómo realiza el programa servidor tanto la lectura como la escritura. La lectura la realiza en unidades cuya longitud aparece indicada en el parámetro *–t*, comentado en la parte de descripción de este programa. Todas la respuestas tienen una longitud fija de **24** octetos.

Independientemente del tamaño del bloque de información que suministre el cliente, el protocolo *TCP* proporciona al programa servidor bloques de información de la longitud solicitada por el servidor que en el caso del ejemplo que estamos viendo es de **256** octetos.

En el inicio de la conexión un participante informa al otro del tamaño de segmento

En el caso de la escritura el *TCP* tiende a enviar el menor número posible de segmentos y ello lo consigue acumulando el máximo número de octetos en el segmento. No obstante las dos primeras respuestas son enviadas en segmentos individuales porque de todas formas es necesario mandar los segmentos de acuse de recibo y aprovechar el envío de estos segmentos para mandar los bloques de información que tenga preparados. La filosofía de este comportamiento es intentar ocupar el menos ancho de banda posible en la línea e intentar entregar la información lo antes posible. Si se suma la longitudes de los datos obtenidos obtiene **192** y como la longitud de los datos de respuesta es **24**, se obtienes **192/24 = 8** respuestas.

El bloque de información que se envía es de **2048** octetos y cada petición de lectura del programa servidor es de **256** octetos, por lo que el programa debe de realizar **8** peticiones de lectura sobre el *socket*. Como cada una de ellas genera un paquete de salida coincide con el número de respuestas obtenidas más arriba. Si se realiza la misma prueba haciendo que el programa servidor lea bloques de **128** octetos se obtendrán **16** respuestas.

Como vemos, el tráfico masivo necesita de mecanismos más complejos como son la segmentación de los datos, cosa que normalmente no se da en el tráfico interactivo. La segmentación de los datos y el máximo tamaño de segmento depende de las características de los equipos que están envueltos en la transmisión y es un eficaz mecanismo que utiliza *TCP* para gestionar la memoria para el manejo de comunicaciones.

Otro problema que presenta el tráfico interactivo es la diferencia de velocidad de proceso entre el receptor y el emisor, sobre todo en el caso de un emisor rápido y un receptor lento, este problema es gestionado por la ventana de transmisión.

UN MECANISMO PARA AJUSTAR: LAS VENTANAS

Las ventanas son un mecanismo de control de flujo mediante el cual cada extremo de la sesión limita al otro extremo el volumen de los datos enviados. Para poder analizar con **tcpdump** el mecanismo de ventanas vamos a poner el intervalo de respuesta a **5** segundos bien arrancando el programa:

```
tcptest -d 7007 -t 256 -i 5
```

o si este está arrancado modificando este valor con el comando:

```
tcpparm -i 5
```

Una vez asignado el intervalo con el valor deseado iniciamos una nueva sesión con **tcpcli** y volvemos una vez más a analizar el establecimiento de la sesión donde podemos observar en los segmentos *SYN* un número que viene a continuación de la palabra *win*. Este número indica el tamaño de la ventana, y este valor recibe el nombre de **tamaño de la ventana**. Para ver cómo se modifica en una transmisión masiva de datos vamos a realizar un envío largo de **10K**. Este bloque de datos enviado es leído por el programa en bloques de **256** octetos y a una velocidad de **1** bloque cada cinco segundos. Los datos que están a la espera de ser leídos por el programa quedan almacenados en un *buffer* de memoria que suministra el *TCP*, pero hasta que estos datos no sean leídos están ocupando espacio en ese *buffer*.

Como el tiempo de respuesta en el servidor es lento se puede observar como al principio según va recibiendo segmentos de **1460** octetos la ventana va disminuyendo hasta que llega a un mínimo cuando el emisor ha mandado toda la información y se mantiene en este valor hasta que se va liberando memoria de *buffer*. En ese momento la ventana vuelve a ir aumentado hasta que lega al máximo valor que puede alcanzar.

Con esta prueba aunque se ha disminuido el tamaño de la ventana todavía es posible enviar más bloques de datos. Para comprobar cómo se cierra la ventana vamos a realizar varios envíos de **10 K**, hasta comprobar que la ventana queda cerrada. Una vez que observamos que el tamaño de la ventana ha llegado a **0** pulsamos un nuevo envío más para asegurarnos que quedan en el cliente datos por enviar.

Se calcula que aproximadamente un 50% de los segmentos TCP son interactivos

Si seguimos la traza de esta prueba observamos como la ventana va reduciendo su tamaño hasta llegar a **0**, lo que significa que el servidor está diciéndole al cliente "tengo los *buffers* de memoria llenos, no me envíes más". A intervalos podemos observar cómo la ventana se abre un poco para inmediatamente volver a cerrarse después de recibir un nuevo segmento de datos de los que quedan pendientes por enviar en el cliente Esta secuencia se mantiene mientras el cliente tenga datos en sus *buffers* por enviar. Una vez que ha sido enviado el último segmento pendiente la ventana vuelve a cerrarse y a partir de este momento y tras un periodo de tiempo en que la ventana está cerrada, ésta empezará a abrirse paulatinamente hasta volver a tomar su máximo tamaño, cuando haya procesado todos los datos.

FLAG PUSH

En casi todos los segmentos de datos se ha visto el *flag PUSH*, así que vamos a describir el significado que tiene este *flag*. Se trata de una indicación del emisor al receptor para que éste pase los datos al proceso que los espera. Estos datos consisten tanto en los que están en el segmento con el *flag PUSH* como los otros datos que pudieran estar pendientes de entregar dentro de los *buffers* del sistema receptor.

Si no se especifica nada en la programación, la mayoría de las implementaciones de *TCP* mandan este *flag* cuando el *buffer* de salida queda vacío.

TEMPORIZADORES

TCP es un protocolo de transmisión que garantiza que los paquetes no se van "perder" durante la transmisión y que estos llegan en el orden adecuado. Para conseguirlo utiliza unos números que mantienen la secuencia, pero este mecanismo no es suficiente y debe ser complementado con los temporizadores. *TCP* tiene cuatro temporizadores distintos para cada conexión:

- Temporizador de transmisión. Se usa cuando se espere un *ACK*.
- Temporizador que mantiene actualizada la información de la ventana.
- Temporizador *keepalive*, que detecta cuándo el otro extremo de la conexión ha finalizado anormalmente sin poder avisar.
- Temporizador *2MSL*. Se utiliza en el cierre de la conexión.

El temporizador de transmisión ya ha sido introducido anteriormente cuando se ha explicado un mecanismo de *ACK* retrasado y se activa cuando se envía un segmento de datos y se espera el *ACK* a dicho segmento. Cuando este temporizador expira, el *TCP* del lado emisor asume que el segmento se ha perdido y lo vuelve a enviar.

El temporizador de ventana es activado por el emisor cuando le llega un *ACK* con la ventana del receptor cerrada. Cuando cumple este intervalo de tiempo el emisor solicita información sobre el tamaño de la ventana. Este temporizador existe para prevenir la situación en la que después de cerrarse la ventana el segmento que abre la ventana se pierde. Si no existiera este temporizador el cliente nunca se enteraría de la apertura de la ventana y la transmisión quedaría interrumpida.

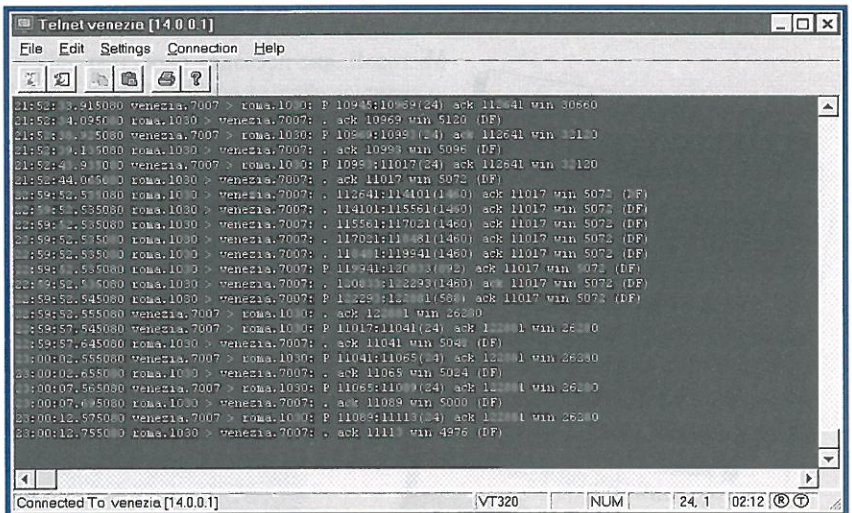


Figura 10. Esquema de la transmisión cerrada cuando el servidor retrasa el tiempo de respuesta.

LA BIBLIA DE ORACLE 8

Gracias a este libro obtendrás un conocimiento firme de Oracle y su línea completa de productos, programas y servicios para su desarrollo y administración. Mejorarás tus técnicas de desarrollo en todas las áreas de Oracle8, desde el diseño hasta la seguridad de las aplicaciones. Del mismo modo conocerás el modo de programar efectivamente con SQL y PL/SQL para obtener así el máximo rendimiento de tu base de datos Oracle. También podrás examinar la teoría que hay detrás de los VLDB, de los LOB de Oracle 8 del almacenamiento de datos.

En definitiva hazte un experto usuario y programador de Oracle 8 gracias a este interesante y atractivo libro.

Incluye un CD-ROM que incorpora dos libros (en inglés) vinculados rápidamente en formato electrónico que son Puesta a punto del rendimiento y optimización de Oracle y Guía de supervivencia del Administrador de la base de datos, DBA, de Oracle. Además están incluidos el código fuente y ejemplos del autor del libro y software de evaluación de terceros entre los que figuran Cristal Reports, SQL Run Demo, CE/2000 y Report Writer SQL Edition entre otros muchos.

Editorial: Anaya
Nº páginas: 918
Nivel: Intermedio-avanzado

Autor: Advanced Information System.
Idioma: Español
Precio: 5.495 Ptas.(I.V.A. inc.)

VISUAL J++ 6.0 MANUAL DEL PROGRAMADOR

Con este manual vas a descubrir cómo las últimas mejoras, de una de las herramientas Java más importantes de la industria, pueden ofrecer grandes beneficios para Windows y la Web. Visual J++ 6.0 combina todo el poder de la plataforma Windows con la productividad del lenguaje Java, creando buenas oportunidades para la ejecución de aplicaciones extendidas.

El libro que necesita para aprender a programar con Microsoft® Visual J++™ 6.0
Microsoft® Visual J++™ 6.0
Manual del programador
Microsoft Press

Este manual te ofrece toda la información, directamente del propio equipo desarrollador de Microsoft, que necesitas para crear aplicaciones sostenidas por las tecnologías ActiveX o sistemas cliente/servidor dirigidos por datos, ayudándote a adquirir todos los conocimientos sobreVisual J++.

Se comentan en este libro todas las novedades de esta última versión como son las Windows Foundation Classes, DHTML3 embebido, procesos cruzados, depuración remota y compilación condicional, así como la implantación de aplicaciones con un sólo botón y muchas otras innovaciones.

También trata sobre temas como son las bases de la programación con Visual J++ incluyendo diseño de formularios, asistentes, edición y depuración de código, así como el acceso a datos, avanzado y seguridad de datos.

Editorial: McGraw-Hill
Nº páginas: 526
Nivel: Intermedio-avanzado

Autor: Microsoft
Idioma: Español
Precio: 7.000 Ptas.(I.V.A. inc.)

LA BIBLIA DE DELPHI 3

La Biblia de Delphi 3 es una versión exhaustivamente revisada y actualizada del libro más afamado sobre programación con Delphi que existe actualmente en el mercado. Escrito por un experto programador de Delphi con reconocimiento internacional, Marco Cantú.

Con el libro podrás estudiar y reutilizar cientos de ejemplos prácticos ejecutables compilados para Delphi 3.

Aprenderás a utilizar en tus programas cualquiera de los objetos Delphi que se construirán a lo largo de los capítulos del libro. Aprende como se utilizan de un modo efectivo las clases, objetos, unidades, métodos e interfaces de Pascal. Asimismo aprovecharás al máximo las ventajas de las funciones visuales de herencia de formularios, el Depósito de Objetos y las cualidades de todos los elementos avanzados y novedosos.

Podrás construir veloces aplicaciones de 32 bits para Windows 95, abordar el desarrollo de los controles ActiveX y ActiveForm, la programación cliente/servidor y mucho más.

Editorial: Anaya
Nº páginas: 678
Nivel: Intermedio-avanzado

Autor: Marco Cantú
Idioma: Español
Precio: 5.995 Ptas.(I.V.A. inc.)

El libro va acompañado de un CD-ROM que contiene el código fuente completo del libro así como numerosos archivos ejecutables.

A FONDO DIRECTX

Este libro es la guía definitiva para trabajar con los avanzados componentes multimedia de alto rendimiento de Microsoft, está escrito por los profesionales que trabajaron con el equipo de DirectX durante el desarrollo de DirectX5.

Contiene toda la información que te muestra cómo utilizar la impresionante potencia multimedia del producto para aumentar el rendimiento y funcionalidad de un amplio rango de aplicaciones.

En el libro puedes encontrar una visión general de DirectX, además de una breve revisión de los fundamentos de Windows, COM y otros conceptos de programación necesarios para trabajar con DirectX. También aborda el tratamiento completo de DirectDraw, DirectSound, DirectPlay, DirectInput y DirectSetup.

A fondo DirectX ahonda en las técnicas de programación multimedia y te muestra cómo obtener la máxima funcionalidad de DirectX. También descubrirás una valiosa fuente de información, hasta ahora no publicada, desde la perspectiva de los creadores, arquitectos y desarrolladores del equipo.

El libro va acompañado por un interesante CD-ROM que contiene el SDK de Microsoft DirectX 5.2 y 6.0.

Editorial: McGraw-Hill
Nº páginas: 500
Nivel: Intermedio-avanzado

Autor: B. Bargen y P. Donnelly
Idioma: Español
Precio: Consultar

Dudas técnicas

En esta sección, los lectores de SÓLO PROGRAMADORES tienen la oportunidad de hallar respuesta a los problemas que puedan tener en cualquier tema relacionado con la programación en cualquier entorno.

Pregunta

Ante todo agradeceros el interés que mes a mes os tomáis elaborando la revista que me parece excelente. Soy un programador 'novato' y me surgen muchas dudas acerca del registro de Windows. En concreto me gustaría saber qué es y qué contiene el apartado *Hkey_Local_Machine*. Además, me imagino que para mi y para muchos, un breve comentario sobre qué es y para qué se usa el registro de W95/98 no nos vendría mal.

Respuesta

Una de las 'lagunas' de los programadores suele ser el registro de W95/98. Intentemos aclarar un poco las cosas.

El registro es una base de datos creada por W95/98 durante la instalación. Las entidades en esta BD son bien software, hardware, usuarios o preferencias sobre una máquina local o de la red. Cada vez que se realiza un cambio en la configuración, automáticamente se refleja en el registro.

Para acceder a este registro usamos un programa denominado *REGEDIT.EXE*. El programa se encuentra en la carpeta *WINDOWS/SYSTEM*.

El registro pretende eliminar los obsoletos ficheros *INI*, sobre todo intentando que la información pueda ser compartida por varias aplicaciones, cosa que indudablemente los ficheros *INI* no podían hacer. Incluso en red, un usuario puede conectarse en cualquier máquina tal y como lo haría en la suya propia. De hecho, varios usuarios pueden almacenar información en el mismo registro.

El editor de registros (*REGEDIT*) nos muestra la información agrupada en seis subramas. Éstas contienen cadenas de texto e incluso otras subramas. Todas comienzan por *HKey* que indica que pueden ser usadas para acceder a recursos.

Expliquemos un poco las seis *Hkey_ handles*

1) **Hkey_Local_Machine**
   Contiene información específica sobre el tipo de hardware, software u otras preferencias en un PC. Esta información es usada por todos los usuarios que se conectan desde esta máquina.
2) **Hkey_Current_Config**
   Apunta a una rama, Key Hkey_Local_Machine \Config, que contiene información sobre la configuración del hardware.
3) **Hkey_Dyn_Data**
   Apunta a una rama de *Hkey_Local_Machine* que contiene información sobre la configuración *Plug and Play* del sistema. Esta información es dinámica y cambia cada vez que un dispositivo se añade al sistema.
4) **Hkey_Classes_Root**
   Apunta a una rama de *Hkey_Local_Machine* que describe ciertas características de software instalado. Incluye información esencial de *OLE* y *DRAG* and *DROP*, accesos directos y aspectos del núcleo de W95/98.
5) **Hkey_Users**
   Ofrece información sobre los usuarios que se conectan al sistema. Cada usuario que se conecta a esta máquina posee su propia *subKey*.
6) **Hkey_Current_User**
   Apunta a una rama de *Hkey_Users* específica del usuario que está actualmente usando la máquina.

Pregunta

Estoy intentando efectuar búsquedas desde *Visual Basic* y la verdad es que no lo tengo muy claro. ¿Podríais echar-

me una mano, con una visión global de los procedimientos y funciones que debo usar? En concreto para los modos *dbOpenDynaset* y *dbOpenSnapShot*. Gracias por vuestra ayuda, y ante todo, por la revista.

Respuesta

Antes de empezar a ver los métodos que se deben utilizar para realizar las búsquedas, vamos a ver los tipos de recordset que podemos crear, puesto que dependiendo del tipo del *recordset* que hayamos creado podremos realizar un tipo de búsquedas u otro. Para asignar un conjunto de registros a un recordset utilizábamos la siguiente sentencia:

```
set SOLOP = baseSOLOP.openrecord-
    set("SOLOP", dbOpenTable)
```

En este caso utilizamos la constante *dbOpenTable*, con lo que le estamos diciendo que asigne al recordset **SOLOP** todos los campos de la tabla **SOLOP**, que deberá ser una tabla de la base de datos **baseSOLOP.**.

Podemos insertar además:

- **dbOpenTable**: Asigna a un recordset el contenido completo de una tabla con todos sus campos y registros. Podemos realizar operaciones de lectura, modificación y escritura sobre los registros. Pero solo de una tabla.
- **dbOpenDynaset**: Se pueden usar varios campos de varias tablas, también permite que se asignen los registros que cumplan ciertas condiciones. Para indicar los campos se utiliza la sentencia *SELECT* . Este método *dbOpenDynaset* permite que se realicen operaciones de lectura, modificación y escritura aunque es más lento que el modo *dbOpenTable*.
- **dbOpenSnapShot**: Abre *recordsets* sólo para lectura. Este modo es más rápido que el anterior y lo utilizaremos cuando sólo nos interese realizar una vista o consulta de un conjunto de tablas. Para asignar el conjunto de registros al recordset utilizaremos también una sentencia *SELECT*.

Las búsquedas varían de un tipo de recordset a otro.

Para los modos *dbOpenDynaset* y *dbOpenSnapShot* usaremos los siguientes métodos:

- **FindFirst**: busca el primer registro que cumple los criterios indicados desde el primer registro del recordset en adelante
- **FindNext**: busca el siguiente registro que cumple los criterios, la búsqueda se realiza desde el último registro encontrado hasta el final del recordset.
- **FindLast**: Similar a FindFirst, pero comienza a realizar la búsqueda desde el final del recordset hacia el principio
- **FindPrevious**: Busca el anterior registro que cumple los criterios, la búsqueda se realiza desde el anterior registro encontrado hasta el principio del recordset.

Su sintaxis es *recordset.findfirst* "cadena de texto con la condición campo operador lógico valor". Una vez que hemos efectuado la búsqueda nos quedará saber si ha habido éxito o no. Esto lo conseguimos comprobando el estado de la propiedad *nomatch* que contiene cualquier recordset.

Si el valor de *nomatch* es **TRUE** significa que no se ha encontrado el registro que buscábamos.

Si el valor de *nomatch* es **FALSE** significa que la búsqueda ha tenido éxito y el *recordset* apuntará al registro que se haya encontrado, con lo que lo tendremos preparado para realizar cualquier operación: modificar, eliminar, mostrar, etc..

Pregunta

Estimados amigos, he decidido enviaros esta consulta, con el fin de podáis ayudarme. Utilizo *PERL* desde hace bastante tiempo y me gustaría que explicaseis con un poco de detalle todos los operadores que podemos encontrar los programadores en este entorno.

Respuesta

PERL soporta una gran cantidad de operadores, la mayoría de ellos heredados del *lenguaje C*. En *PERL* los operadores conservan el mismo uso que en el resto de lenguajes, y se utilizan para realizar operaciones aritméticas, operaciones lógicas ..., entre las variables del programa. Los operadores pueden clasificarse en los siguientes tipos:

Operadores de comparación: mediante estos operadores podemos enfrentar dos expresiones, de manera, que dependiendo de su valor se generará un resultado que puede ser lógico o numérico. PERL tiene los siguientes operadores de comparación:

- == . Este operador sirve para comparar dos valores iguales, de manera que cuando las expresiones a su izquierda y su derecha son iguales, devuelve un valor lógico verdadero y cuando no lo son devuelve falso.
- **eq** . Es empleado para comparar expresiones no numéricas, esto es, se utiliza de igual manera que == pero para cadenas.
- != . Con este operador se comparan cantidades numéricas diferentes.
- **ne** . Se utiliza para comparar cantidades no numéricas diferentes. Su funcionamiento es similar que el de != .
- < . Verifica el valor de una cantidad numérica con respecto del valor de la expresión de la derecha, de modo que si el valor de esta expresión es

DB2 y AIX son marcas registradas de IBM Corp. Se reconocen todas las marcas registradas. © 1999 IBM.

No haga esperar a sus Clientes. Con DB2 Universal Database, todos podrán ser el primero de la fila.

Porque DB2 Universal Database es la base de datos diseñada para hacer negocios a través de Internet.

Completamente compatible con Java, proporciona soporte para textos, gráficos, sonido y vídeo para poner en marcha aplicaciones Self-Service en Internet.

Y todo ello simultáneamente y al instante para miles de usuarios, de modo que todos sean los primeros y reciban la mejor atención sin tener que esperar colas.

Además, sus completas funciones de conectividad facilitan la consolidación de datos provenientes de cualquier fuente y en cualquier plataforma. Y, a propósito de plataformas, DB2 Universal Database funciona en modo nativo en una gran cantidad de ellas, desde Windows NT hasta Sun Solaris y AIX.

Visítenos en www.software.ibm.com/webservers/websoftware/sp

e-business

Consiga sin cargo alguno un kit de iniciación del software Web Self-Service de IBM que incluye un CD de demostración de DB2 Universal Database en www.software.ibm.com/webservers/websoftware/sp

IBM

Soluciones para nuestro pequeño mundo